Über die Sicherheit und Effizienz kryptographischer Verfahren in algebraischen Zahlkörpern

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Inform. Andreas Alfred Meyer

aus Saarlouis

Gutachter:	Prof. Dr. Johannes Buchmann
	Prof. Dr. Tsuyoshi Takagi
Tag der Einreichung:	24.10.2005
Tag der mündlichen Prüfung:	05.01.2006

Darmstadt, 2005
D 17

Über die Sicherheit und Effizienz kryptographischer Verfahren in algebraischen Zahlkörpern

Dipl.-Inform. Andreas Alfred Meyer

Technische Universität Darmstadt
Fachbereich Informatik
Hochschulstraße 10
D–64289 Darmstadt

Mathematics Subject Classification (2000): 11R04, 11R29, 11T71, 11Y40, 68P25, 94A60
ACM Computing Classification System (1998): E.3, F.2.1

Danksagung

An dieser Stelle danke ich meinen Eltern und meiner Frau Marcela für die moralische Unterstützung bei der Erstellung meiner Dissertation. Diese war insbesondere in den Zeiten notwendig, in denen mich meine berufliche Tätigkeit in der Industrie zeitlich in besonderem Maße beansprucht hat. Die zahlreichen fachlichen Gespräche mit meinem Vater über neue Teilergebnisse machten große Freude und ermutigten mich in besonderer Weise, nach weiteren neuen Problemlösungen zu forschen.

Mein herzlicher Dank gilt Herrn Prof. Dr. Johannes Buchmann, von dem ich zunächst als Student, dann als Doktorand vieles gelernt habe. Die Diskussionen mit Herrn Buchmann haben mir stets weiter geholfen, unabhängig davon, ob ihr Gegenstand Mathematik und Kryptographie, meine beruflichen Aktivitäten in der Industrie oder Privates war. Herrn Prof. Dr. Tsuyoshi Takagi danke ich für dessen Bereitschaft, als Zweitgutachter zu fungieren.

Der Deutschen Forschungsgemeinschaft danke ich für die finanzielle Unterstützung, die ich zeitweise für meine Forschungsarbeiten erhielt. Ich danke meinen Kollegen Tobias Hahn, Dr. Safuat Hamdy, Andreas Petter, Thomas Pfahler und insbesondere Dr. Stefan Neis für die Vielzahl interessanter Gespräche und Anregungen zu meiner Arbeit.

Zu besonderem Dank bin ich meinem Förderer und Freund Ismet Koyun verpflichtet, der mir gemeinsam mit Herrn Prof. Buchmann die einmalige Chance eröffnet hat, neben der Erstellung einer akademischen Forschungsarbeit gleichzeitig praktische Erfahrungen in der Entwicklung und internationalen Vermarktung kryptographischer Verfahren in Form von IT-Sicherheitsprodukten zu sammeln. Ich denke, Herr Koyun hat recht behalten in seiner vor etlichen Jahren geäußerten Prognose, ich werde in seiner Firma KOBIL einen zweiten (virtuellen) Doktor machen; in den vergangenen Jahren konnte ich in der engen Zusammenarbeit mit Herrn Koyun vieles lernen und in verantwortlicher Position zunächst in der Entwicklung und im Produktmanagement, dann in Vertrieb und Marketing zum Unternehmenserfolg beitragen.

Erklärung[1]

Hiermit erkläre ich, dass ich die vorliegende Arbeit - abgesehen von den in ihr ausdrücklich genannten Hilfen - selbständig verfasst habe.

Wissenschaftlicher Werdegang des Verfassers in Kurzfassung[2]

10/1992 bis 12/1997	Studium der Informatik mit Nebenfach Mathematik an der Universität des Saarlandes
10/1994	Vordiplom mit der Gesamtnote "sehr gut"
12/1997	Diplom mit der Gesamtnote "sehr gut" Studienschwerpunkte: Kryptogrpahie und algorithmische Zahlentheorie Diplomarbeit: "Ein neues Identifikations- und Signaturverfahren über imaginär-quadratischen Klassengruppen
11/1998 bis 11/2002	wissenschaftlicher Mitarbeiter bei Prof. Dr. J. Buchmann, Fachbereich Informatik an der Technischen Universität Darmstadt

[1] gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt
[2] gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt

Zusammenfassung

In dieser Arbeit leisten wir einen Beitrag zur Suche nach alternativen sicheren und effizienten Kryptoverfahren. Wir befassen uns mit asymmetrischen Kryptoverfahren, deren Sicherheit auf der Schwierigkeit des diskreten Logarithmusproblems, des Diffie-Hellman-Problems oder des Wurzelproblems in endlichen Abelschen Gruppen beruht. Die Kenntnis der Gruppenordnung wird hierbei nicht benötigt. Daher lassen sich die Kryptoverfahren in Klassengruppen algebraischer Zahlkörper (number fields) vom Grad größer 2 sicher implementieren. (In Zahlkörpern implementierte Verfahren nennen wir *NF-Kryptoverfahren*.)

Bislang sind über Kryptographie in algebraischen Zahlkörpern (NF-Kryptographie) erst zwei Arbeiten erschienen. In [Buc91] wurde erstmals die Idee formuliert, Kryptographie in algebraischen Zahlkörpern zu betreiben. Diese Idee wurde in [BP97] konkretisiert, indem ein für kryptographische Verfahren scheinbar geeignetes mathematisches Basisproblem beschrieben wurde. Das Lösen dieses Basisproblems erscheint schwierig, selbst wenn die heute in der Praxis in Kryptoverfahren verwendeten Basisprobleme gelöst würden, z.B. das Faktorisierungproblem ganzer Zahlen oder das diskrete Logarithmusproblem in endlichen Körpern oder der Punktegruppe einer elliptischen Kurve über endlichen Körpern. Daher scheint dieses neue Basisproblem eine attraktive Alternative zu heute verwendeten Basisproblemen zu sein.

Andere publizierte Arbeiten bezogen sich bislang nur auf den Spezialfall quadratischer Zahlkörper (siehe z.B. [BW88, BDW90, McC89, BD90, Dül91, Mey97, HJP98, HMT98, HM00, BH01, Tak01, BW89, BSW90, BSW94, BBT94, BW90]).

Die mathematischen Basisprobleme, die zum Brechen der Kryptoverfahren gelöst werden müssen, sind für Zahlkörper vom Grad größer 2 noch schwieriger als für quadratische Zahlkörper. Das macht Zahlkörper höheren Grades für die Kryptographie besonders interessant.

Bisher waren noch etliche Fragestellungen bzgl. NF-Kryptographie offen: Es war unklar, welche Signaturverfahren für Klassengruppen algebraischer Zahlkörper verwendet werden können, denn die Ordnung der Klassengruppe oder eines Teilers davon ist im Allgemeinen nicht effizient berechenbar. Daher können Signaturen vom ElGamal-Typ (z. B. DSA) nicht ohne weiteres auf Klassengruppen übertragen werden, da für Signaturen dieses Typs die Gruppenordnung bekannt sein muss. Zudem wurde bislang nicht nachgewiesen, dass man neben Signaturverfahren andere kryptographische Anwendungen sicher und effizient in algebraischen Zahlkörpern implementieren kann. Weiterhin war vollkommen unklar, wie man kryptographisch geeignete Klassengruppen algebraischer Zahlkörper effizient erzeugen sollte. Denn im Allgemeinen sind solche Klassengruppen sehr klein, während wir große Klassengruppen benötigen. Zudem muss die Gruppenordnung große Primteiler enthalten. Wie soll man aber die Existenz großer Primteiler in einer vollkommen unbekannten Gruppenordnung nachweisen? Es war unklar, wie in der Praxis effizient entschieden werden soll, ob zwei gegebene Gruppenelemente (Idealklassen) gleich sind. Letztendlich existierte noch keine Implementierung von NF-Kryptoverfahren; somit war vollkommen offen, ob NF-Kryptoverfahren überhaupt hinreichend effizient implementiert werden können. Über die Schwierigkeit der mathematischen Basisprobleme gab es bislang nur grobe theoretische, asymptotische Abschätzungen, aber noch keine praktischen Untersuchungen. Daher wusste man bislang nicht, wie groß die Systemparameter von NF-Kryptoverfahren zu wählen sind, um ein gewünschtes Sicherheitsniveau zu erreichen.

Mit dieser Arbeit tragen wir zur Klärung dieser bislang offenen Fragen bei: Wir zeigen, dass man überhaupt kryptographische Protokolle über Klassengruppen algebraischer Zahlkörper entwerfen kann. Hierzu stellen wir einige NF-Kryptoverfahren zur Signatur, zur Verschlüsselung, zum Schlüsselaustausch, zur Identifikation und für weitere Anwendungen vor.

Wir zeigen, dass die NF-Kryptoverfahren sicher sind unter der Annahme, dass die Berechnung diskreter Logarithmen und Wurzeln in Klassengruppen nicht effizient möglich ist. Wir erläutern, wie der algebraische Zahlkörper ausgewählt werden soll, damit diese Annahme nach heutigem Kenntnisstand mit sehr hoher Wahrscheinlichkeit erfüllt ist. Mittels theoretischer Resultate und praktischer Untersuchungen zeigen wir ferner, wie groß die Parameter (die Diskriminante und der Grad eines Zahlkörpers) ausgewählt werden müssen, damit das Brechen der NF-Kryptoverfahren nach heutigem Kenntnisstand genauso schwierig ist wie das Brechen heute in der Praxis verwendeter Kryptoverfahren wie RSA (mit z.B. 1024 Bit Schlüssellänge).

Zudem untersuchen wir die Effizienz der NF-Kryptoverfahren und der darunter liegenden Arithmetik für Klassengruppen algebraischer Zahlkörper. Wir zeigen, dass NF-Kryptoverfahren schon hinreichend effizient sind, um auf modernen Computern eingesetzt werden zu können.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Sicherheit der meisten gängigen kryptographischen Verfahren beruht auf der Schwierigkeit der Faktorisierung natürlicher Zahlen oder der Berechnung diskreter Logarithmen (DL) in endlichen Körpern oder in der Punktegruppe elliptischer Kurven über endlichen Körpern. Es ist aber absolut unklar, ob diese Probleme in der Zukunft schwierig bleiben. Im Gegenteil. In den letzten 15 Jahren gab es bei der Entwicklung effizienter Faktorisierungs- und diskreter Logarithmusalgorithmen in endlichen Körpern sehr große Erfolge ([LV99, LV01]). Fortschritte bei der Lösung des Faktorisierungsproblems führten in der Regel auch zu schnelleren DL-Algorithmen in endlichen Körpern. Darüber hinaus fanden Mathematiker immer wieder Algorithmen, welche das diskrete Logarithmusproblem für Familien von elliptischen Kurven über endlichen Körpern so effizient lösen, dass diese für die Kryptographie nutzlos werden ([MOV91, FR94, Sma99, Sma00]). Nehmen wir an, man fände entsprechend schnelle Algorithmen für die o.g. algorithmischen Probleme. Kryptographische Verfahren wie RSA oder DSA wären dann auf einen Schlag unsicher. Diese Kryptoverfahren sind heute aber in realen Anwendungen wie E-Business-Applikationen, Remote Access-Anwendungen oder Absicherung digitaler Kommunikation weit verbreitet. Daher entstünde für die Industrie weltweit ein gigantischer wirtschaftlicher Schaden auf Grund der dann einfach durchführbaren Wirtschaftsspionage oder Betrugsvergehen. Zudem müssten gängige digitale Kommunikationskanäle abgeschaltet werden, da diese nun nicht mehr hinreichend abgesichert werden könnten.

Daher besteht eins der Hauptziele der heutigen Public-Key-Kryptographie darin, neue Berechnungsprobleme zu identifizieren und auf deren Basis sichere und effiziente Public-Key-Kryptoverfahren zu konstruieren.

In der vorliegenden Arbeit wollen wir zeigen, dass das algorithmische Problem, diskrete Logarithmen (NF-DLP) oder Wurzeln (NF-RP) in der Klassengruppe geeigneter algebraischer Zahlkörper zu berechnen, Grundlage für die Sicherheit effizienter Kryptosysteme sein kann. Die genannten Probleme NF-DLP und NF-RP sind deshalb von besonderem Interesse, weil die bekannten Algorithmen zu deren Lösung die Berechnung kürzester Vektoren in Gittern der Dimension n erfordern, wobei n der Zahlkörpergrad ist. Die derzeit besten Algorithmen zur Berechnung kürzester Vektoren in solchen Gittern haben exponentielle Laufzeit in n. Man kann außerdem zeigen, dass NF-DLP wenigstens so schwer ist wie das Faktorisierungsproblem für natürliche Zahlen (siehe Abschnitt 3.1). Für das Wurzelproblem in Klassengruppen algebraischer

Zahlkörper kennt man heute keine effizienteren Algorithmen als für das NF-DLP. Die Probleme NF-RP und NF-DLP erweitern also den Kreis der schweren zahlentheoretischen Probleme substanziell.

In dieser Arbeit verfolgen wir zwei Ziele:

1. Die Entwicklung effizienter kryptographischer Protokolle, deren Sicherheit auf der Schwierigkeit des NF-DLP oder des Wurzelproblems beruht.

2. Die Untersuchung der Sicherheit entsprechender kryptographischer Verfahren.

Bis jetzt war unklar, ob man überhaupt Kryptographie über Klassengruppen algebraischer Zahlkörper betreiben kann. Die Sicherheit entsprechender kryptographischer Verfahren war demnach bislang auch nicht untersucht worden. Implementierungen existierten bisher nicht; daher gab es keine Erfahrungswerte bzgl. der Effizienz. Bei der Verfolgung obiger Ziele hat man u.a. folgende Probleme zu lösen:

- Entwurf kryptographischer Protokolle
  Klassengruppen algebraischer Zahlkörper sind endliche Abelsche Gruppen, deren Gruppenordnung (Klassenzahl) unbekannt ist und die in der Praxis auch nicht berechnet werden kann. Bis dato war es ein offenes Problem, ob und wie man in solchen Gruppen die gängigen Typen kryptographischer Protokolle konstruieren kann; denn die üblichen Verfahren setzen in der Regel die Kenntnis der Gruppenordnung voraus.

- Erzeugen schwieriger Probleminstanzen
  Bislang wusste niemand, welche notwendigen und hinreichenden Bedingungen an Klassengruppen algebraischer Zahlkörper zu stellen sind, damit über diesen Gruppen erzeugte diskrete Logarithmus- und Wurzelprobleme schwierig zu lösen sind. Als offensichtliche Bedingung fordern wir hier eine sehr große Klassenzahl ein. Allerdings besitzen Klassengruppen algebraischer Zahlkörper im Allgemeinen eine sehr kleine Klassenzahl. Es war vollkommen unklar, wie man Klassengruppen mit den geforderten Eigenschaften konstruieren kann. Schließlich wusste man bislang auch nicht, welche Schlüssellängen zu wählen sind, um bei den neuen kryptographischen Verfahren ein gewünschtes Sicherheitsniveau zu erhalten.

- Effiziente Implementierung
  Bislang existierte keine Implementierung eines kryptographischen Protokolls über algebraischen Zahlkörpern vom Grad > 2. Es war noch nicht hinreichend untersucht, wie man die mathematischen Objekte platzeffizient im Rechner darstellen kann. Außerdem war noch nicht klar, wie man in Klassengruppen am Effizientesten die Gruppenoperation durchführt, Elemente invertiert und effizient entscheidet, ob zwei gegebene Gruppenelemente gleich sind.

1.2 Stand der Forschung

In der Praxis werden zur Zeit folgende Probleme der algorithmischen Zahlentheorie als Grundlage der Sicherheit kryptographischer Verfahren verwendet: die Faktorisierung natürlicher Zahlen und die Berechnung diskreter Logarithmen in der multiplikativen Gruppe eines endlichen Körpers und in der Punktegruppe einer elliptischen Kurve über einem endlichen Körper.

Es gibt subexponentielle Algorithmen zur Faktorisierung natürlicher Zahlen ([BLP93]) und zur Berechnung diskreter Logarithmen in endlichen Körpern ([Sch93]). Genauer gilt folgendes. Setzt man

$$L_x[a,b] = \exp(b(\ln x)^a(\ln\ln x)^{1-a})$$

für reelle Zahlen x, a, b mit $x > e$, so kann man natürliche Zahlen n in erwarteter Laufzeit $L_n[1/3, \left(\frac{64}{9}\right)^{\frac{1}{3}} + o(1)]$ faktorisieren und in endlichen Primkörpern der Charakteristik p diskrete Logarithmen in Zeit $L_p[1/3, \left(\frac{64}{9}\right)^{\frac{1}{3}} + o(1)]$ berechnen. Die derzeit besten Algorithmen zur Berechnung diskreter Logarithmen in der Punktegruppe elliptischer Kurven sind die Algorithmen, die in jeder endlichen Abelschen Gruppe anwendbar sind ([BJT97]). Sie haben exponentielle Laufzeit.

Die Faktorisierung einer Zahl mit 130 Dezimalstellen, die Produkt zweier etwa gleich großer Primzahlen ist, dauert mit dem derzeit besten Algorithmus, dem Number Field Sieve (NFS), ca. 500 MIPS-Jahre ([CDEH$^+$96]). Bei je drei weiteren Dezimalstellen der zu faktorisierenden Zahl verdoppelt sich die Laufzeit des Algorithmus. Die Berechnung des diskreten Logarithmus in einem Primkörper mit 85-stelliger Charakteristik dauert mit dem derzeit asymptotisch besten Algorithmus, dem allgemeinen Number Field Sieve, 44.5 MIPS-Jahre und mit einer Index Calculus–Methode nach Coppersmith, Odlyzko und Schroeppel ([COS86]) 30.6 MIPS-Jahre ([Sch93, DSW96, DM96, Web95, Web96, Web97b, Web97a]). Mit wachsender Charakteristik nimmt die Laufzeit wie beim Faktorisieren zu. Mit dem Algorithmus von Pollard ([Pol75]) wurden in der Punktegruppe einer elliptischen Kurve über einem Primkörper mit 19-stelliger Charakteristik und 18-stelligem größtem Primteiler der Gruppenordnung diskrete Logarithmen auf einer Workstation Sun Sparc 20 in etwa 4 Tagen und 15 Stunden berechnet ([PW97]).

In jüngerer Vergangenheit wird auch untersucht, ob man Jacobische Varietäten hyperelliptischer Kurven über endlichen Körpern in Kryptosystemen verwenden soll ([Kob90, Kob89, Pau96]). Das Problem, kürzeste und nächste Gittervektoren zu berechnen, findet neuerdings auch Anwendung in der Kryptographie ([GGH97]). Es ist aber bis jetzt noch nicht klar, ob diese Verfahren überhaupt effizient implementiert werden können. Außerdem ist noch ungeklärt, wie die Gitter in geeigneter Weise ausgesucht werden können. In den genannten Versuchen spielt die algebraische Zahlentheorie keine Rolle. Das asymptotisch schnellste Verfahren zur Berechnung kürzester Gittervektoren stammt von Bettina Just ([Hel85]), die das Verfahren von Kannan ([Kan87]) verbesserte. Eine genaue Untersuchung von Gitterbasis-Reduktionsalgorithmen wurde von Schnorr ([Sch87]) durchgeführt.

Buchmann und Paulus führten in [BP97] das folgende DL-ähnliche Problem ein: Gegeben seien eine Ordnung $\mathcal{O}$ eines algebraischen Zahlkörpers K und $\mathcal{O}$–Ideale I und J. Finde $k \in \mathbb{Z}_{>0}$ und $\alpha \in K$ mit $J = \alpha I^k$ oder entscheide, dass solche α, k nicht existieren.

In verschiedenen Arbeiten wurde dargelegt, wie man obiges DL-ähnliche Problem für imaginärquadratische und reell-quadratische Zahlkörper als Grundlage von Kryptosystemen verwenden kann ([BW88, BDW90, BW89, BW90, BSW94, BBT94]). Es wurde außerdem gezeigt, dass das Faktorisierungsproblem ganzer Zahlen auf dieses Problem in quadratischen Zahlkörpern reduziert werden kann ([BW88])[1]. Die Idee, Kryptographie basierend auf dem diskreten Logarithmusproblem in Klassengruppen algebraischer Zahlkörper (NF-DLP) zu betreiben, wurde erstmals in [Buc91] vorgestellt.

[1] In imaginär-quadratischen Zahlkörpern ist dieses Problem genau das diskrete Logarithmusproblem, da sich die Releativerzeuger hier einfach berechnen lassen

In [BD90] wurde bewiesen, dass NF-DLP in imaginärquadratischen Klassengruppen in subexponentieller Zeit (in der binären Länge der Diskriminante Δ) lösbar ist. Vollmer [Vol00] bewies hierfür die Komplexität $L_{|\Delta|}[\frac{1}{2}, \frac{3}{4}\sqrt{2}+o(1)]$. Die Lösung von NF-DLP (und des allgemeineren oben formulierten Problems) steht in engem Zusammenhang mit der Berechnung der Klassen- und Einheitengruppe algebraischer Zahlkörper. In [Buc89] wurde unter gewissen Annahmen bewiesen, dass die Klassen- und Einheitengruppe von Zahlkörpern von festem Grad in subexponentieller Zeit in $\log|\Delta|$ berechnet werden kann. Über Erfahrungen bzgl. der praktischen Effizienz dieses Algorithmus berichtet Neis [Nei02]. In [Thi95] wurde ein Algorithmus zur Lösung dieses Problems vorgestellt, dessen Laufzeit exponentiell mit dem Grad des Zahlkörpers wächst. In diesem Algorithmus müssen kürzeste Vektoren in Gittern berechnet werden, deren Dimension der Zahlkörpergrad ist.

Neben den Untersuchungen zur Lösung von NF-DLP und zur Berechnung von Invarianten in Zahlkörpern wurden auch Arbeiten zur Arithmetik in Zahlkörpern erstellt. Insbesondere gab es erste Untersuchungen darüber, wie man die algebraischen Objekte (Ordnungen, Ideale, Körperelemente) geeignet darstellen kann ([Nei94, BTW95, Thi95, Nei02]).

1.3 Aufbau der Arbeit

In Abschnitt 1.4 erklären wir zunächst die in unserer Arbeit verwendeten Notationen.

In Kapitel 2 führen wir die mathematischen Strukturen und Objekte ein, in denen wir Kryptographie betreiben wollen, Klassengruppen algebraischer Zahlkörper. Wir werden sehen, dass Klassengruppen endliche Abelsche Gruppen sind, die mitsamt ihrer Elemente effizient im Computer dargestellt werden können und in denen effizient gerechnet werden kann: Es können effizient Gruppenelemente verknüpft (multipliziert) und invertiert werden. Als Besonderheiten dieser Gruppen werden wir sehen, dass die Entscheidung, ob zwei gegebene Gruppenelemente gleich sind, nur in einigen dieser Gruppen (nämlich in denen mit sehr kleinem Regulator) effizient getroffen werden kann. Zudem ist die Ordnung dieser Gruppen (Klassenzahl) unbekannt. Die Berechnung der Klassenzahl erweist sich als schwieriges algorithmisches Problem, für dessen Lösung nur subexponentielle Algorithmen bekannt sind. Unklar ist zunächst, wie zufällige Elemente in der Klassengruppe ausgewählt werden können, was zur Implementierung kryptographischer Protokolle aber notwendig ist. All diese arithmetisch-algorithmischen Probleme werden wir detaillierter in den Kapiteln 4 und 5 betrachten.

In Kapitel 3 zeigen wir, dass sich eine Fülle kryptographischer Protokolle in Klassengruppen algebraischer Zahlkörper implementieren lassen. (Bei Implementierung in Klassengruppen nennen wir die Protokolle *NF-Kryptoverfahren*.) Wir stellen entsprechende Protokolle vor, erläutern ggf. deren Einsatzgebiete, analysieren deren Sicherheit und Effizienz und geben schließlich Empfehlungen zur Parameterwahl. Wir zeigen durch Konstruktion entsprechender Protokolle, dass sich quasi alle gängigen Typen kryptographischer Protokolle in Klassengruppen algebraischer Zahlkörper formulieren lassen. Wir stellen in Kapitel 3 zunächst die algorithmischen Probleme vor, auf deren Schwierigkeit die Sicherheit der von uns vorgestellten Verfahren basiert. Wir erklären durch Aufzeigen von Polynomzeitreduktionen, in welchem Zusammenhang diese algorithmischen Probleme stehen und wie schwierig diese nach heutigem Kenntnisstand sind. Anschließend modellieren wir den Begriff der Sicherheit und definieren, was wir darunter verstehen, dass ein kryptographisches Protokoll „sicher“ sei. Dann analysieren wir teilweise vollkommen neue kryptographische Protokolle, die sich in Klassengruppen algebraischer Zahlkörper implementieren lassen: Signaturverfahren

mit der üblichen Funktionalität, Nichtabstreitbare Signaturen (Undeniable Signatures), Blinde Signaturverfahren (Blind Signatures), Gruppensignaturen; Verschlüsselungsverfahren, Schlüsselaustauschverfahren, Identifikationsverfahren, Faire Public-Key-Kryptosysteme, Pseudozufallszahlengeneratoren, Bit Commitment und fairer Münzwurf, Secret Sharing und Subliminal Channel.

In Kapitel 4 analysieren wir zunächst die Effizienz der bekannten generischen und Klassengruppen-spezifischen Angriffe auf die kryptographischen Protokolle. Diese Angriffe brechen die NF-Kryptoverfahren durch Lösen der zugrunde liegenden algorithmischen Probleme. Insbesondere untersuchen wir die Laufzeit von Buchmanns Klassengruppenalgorithmus und verbessern bisherige Komplexitätsresultate. Wir leiten notwendige und hinreichende Bedingungen dafür her, dass die Angriffe in der Praxis versagen. Daraus resultieren die Anforderungen an eine kryptographisch geeignete Klassengruppe. Anschließend zeigen wir, wie man kryptographisch geeignete Klassengruppen erzeugen kann. Wir weisen nach, dass diese Gruppen tatsächlich allen Anforderungen genügen. Insbesondere zeigen wir hierzu an Hand theoretischer und praktischer Analysen, dass die von uns vorgeschlagenen Klassengruppen große Primteiler enthalten. Zudem untersuchen wir, welche Schlüssellängen man wählen muss, um ein gewünschtes Sicherheitsniveau (z.B. vergleichbar mit RSA-1024 Bit) zu erhalten.

In Kapitel 5 beschreiben wir unsere Implementierung der NF-Kryptoverfahren. Dabei zeigen wir auf, wie man die bislang bekannte (in Kapitel 2 geschilderte) Arithmetik in algebraischen Zahlkörpern optimieren kann. Wir schildern zunächst, wie wir die mathematischen Objekte platzeffizient im Computer darstellen; dann stellen wir effiziente Algorithmen vor und widmen uns insbesondere den nichttrivialen Fragen, wie man entscheiden kann, ob zwei Gruppenelemente gleich sind und wie man zufällig und gleichverteilt ein Element aus der Klassengruppe eines Zahlkörpers auswählen kann. Wir schildern Details unserer Implementierung wie hocheffiziente Methoden zur Exponentiation von Idealklassen. Schließlich präsentieren wir die Laufzeiten unserer neuen Algorithmen und NF-Kryptoverfahren.

Im Anhang leiten wir die in Kapitel 4 zitierten oberen Schranken für Regulatoren und unteren Schranken für Klassenzahlen her. Diese Analyse ist wesentlich, um die praktische Durchführbarkeit und die Sicherheit von NF-Kryptoverfahren zu gewährleisten. Anschließend präsentieren wir Tabellen mit von uns errechneten Klassenzahlen und deren Primfaktorzerlegungen. Die Tabellen machen plausibel, dass die von uns in Kapitel 4 vorgeschlagenen Klassengruppen tasächlich allen Bedingungen genügen.

1.4 Notation

Nachstehend listen wir alle Symbole auf, die wir verwenden werden:

$\mathbb{Z}$	Ring der ganzen Zahlen
$\mathbb{Z}_{\geqslant n}$	Menge der ganzen Zahlen $\geqslant n$ für eine ganze Zahl n
$\mathbb{Q}$	Körper der rationalen Zahlen
$\mathbb{R}$	Körper der reellen Zahlen
$\mathbb{R}_{\geqslant n}$	Menge der reellen Zahlen $\geqslant n$ für eine reelle Zahl n
$\mathbb{C}$	Körper der komplexen Zahlen
K	algebraischer Zahlkörper
$p \mid a$	p teilt a

$p^e \| a$	$p^e \mid a$ aber $p^{e+1} \nmid a$
$(p)_\alpha$	$= \prod_{1 \leqslant k \leqslant \alpha} \left(1 - p^{-k}\right)$
$(p)_\infty$	$= \prod_{k \geqslant 1} \left(1 - p^{-k}\right)$
$\mathrm{Re}\, z$	Realteil einer komplexen Zahl z
$\mathrm{Im}\, z$	Imaginärteil einer komplexen Zahl z
$\lfloor x \rfloor$	die größte ganze Zahl z mit $z \leqslant x$ ($x \in \mathbb{R}$)
$\lceil x \rceil$	die kleinste ganze Zahl z mit $x \leqslant z$ ($x \in \mathbb{R}$)
$\mathrm{size}(n)$	$\left\lfloor \log_2 \lvert n \rvert \right\rfloor + 1$, die Bitlänge von n, wobei $\mathrm{size}(0) = 1$ ist
$\zeta(s)$	Riemannsche Zeta-Funktion
$\theta(x)$	Tschebyschevsche theta-Funktion
$\pi(x)$	Anzahl der Primzahlen $\leqslant x$
$\varphi(n)$	Eulersche Totient-Funktion
$\rho(u)$	Dickmannsche rho-Funktion
$\log$	Logarithmus zur Basis 2
$\ln$	natürlicher Logarithmus
Δ	Diskriminante eines algebraischen Zahlkörpers
$\mathcal{O}, \mathcal{O}_\Delta$	Ordnung eines algebraischen Zahlkörpers (zur Diskriminante Δ)
$Cl(K), Cl(\Delta)$	Klassengruppe des Zahlkörpers K bzw. zur Diskriminante Δ
$R(K), R$	Regulator des Zahlkörpers K
$h(K), h$	Klassenzahl von $Cl(K)$, $h(K) = \lvert Cl(K) \rvert$
$1_{Cl(K)}$	neutrales Element von $Cl(K)$
H_K	Untergruppe von $Cl(K)$, die alle Elemente mit Ordnungen z enthält, wobei $z \in \mathbb{Z}$ beliebig mit $3 \nmid z$ (der Nicht-3-Anteil von $Cl(K)$).
$\mathfrak{a}$, $\mathfrak{b}$	Ideale aus $\mathcal{O}_\Delta$
$[\mathfrak{a}], [\mathfrak{b}]$	Ideal-Klassen in $Cl(K)$ bzw. $Cl(\Delta)$
$\mathrm{ord}_{Cl(\Delta}([\mathfrak{a}])$	Ordnung der Untergruppe von $Cl(\Delta)$, die von $[\mathfrak{a}]$ erzeugt wird
G	endliche Abelsche Gruppe
$\alpha, \beta, \gamma, \ldots$	Elemente einer endlichen Abelschen Gruppe (bisweilen auch Elemente eines algebraischen Zahlkörpers)
$\langle \alpha \rangle$	die von α erzeugte Untergruppe in G
$\mathrm{ord}\, \alpha$, $\mathrm{ord}_G\, \alpha$	Ordnung des Elementes α in G
1_G, 1	neutrales Element in G
$g = O(f)$	Für die Funktionen $g, f : \mathbb{Z}_{\geqslant 0} \longrightarrow \mathbb{R}_{>0}$ gibt es eine reelle Konstante $c > 0$, so dass mit einer positiven ganzen Zahl n_0 für alle ganzen Zahlen $n \geqslant n_0$ gilt: $g(n) \leqslant c \cdot f(n)$ Intuitiv gesprochen: $g(n)$ wächst für wachsendes n asymptotisch höchstens so schnell wie $f(n)$.
$g = o(f)$	Für die Funktionen $g, f : \mathbb{Z}_{\geqslant 0} \longrightarrow \mathbb{R}_{>0}$ gibt es für jede reelle

	Konstante $c > 0$ eine positive ganze Zahl n_0, so dass für alle ganzen Zahlen $n \geqslant n_0$ gilt: $0 \leqslant g(n) < c \cdot f(n)$ Intuitiv gesprochen: $g(n)$ kann in Bezug auf $f(n)$ vernachlässigt werden, wenn n hinreichend groß wird. Der Ausdruck $o(1)$ drückt eine Funktion $g(n)$ aus, die für $n \to \infty$ gegen 0 geht.
$L(N)[u, v + o(1)]$	$e^{(v+o(1))\ln(N)^u \ln(\ln(N))^{1-u}}$ $(0 \leqslant u \leqslant 1)$ Diese Funktion wird benutzt, um die in der Eingabelänge von N subexponentielle Laufzeit eines Algorithmus zu beschreiben.
1 MIPS	In der Kryptographie übliche Maßeinheit für die Rechengeschwindigkeit eines Computers oder einer CPU. Eine 1 MIPS-CPU führt eine Million Operationen pro Sekunde aus.
1 MIPS-Jahr	Anzahl der Operationen, die eine 1 MIPS-CPU in einem Jahr ausführen kann, also $3.1535 \cdot 10^{13}$ Operationen.
$\mathbf{s}_1 \| \mathbf{s}_2$	die Konkatenierung der Bit-Strings $\mathbf{s}_1$ und $\mathbf{s}_2$
$\mathbf{s}_1 \oplus \mathbf{s}_2$	bitweises XOR der Bit-Strings $\mathbf{s}_1$ und $\mathbf{s}_2$

Die verallgemeinerte Riemannsche Vermutung (GRH)

Die *verallgemeinerte Riemannsche Vermutung (GRH)* ist eine allgemein als wahr angenommene mathematische Behauptung, die aber bislang nicht bewiesen werden konnte. Wir werden einige Aussagen herleiten, deren Beweis voraussetzt, dass die verallgemeinerte Riemannsche Vermutung korrekt ist.

In der gewöhnlichen Riemannschen Vermutung wird angenommen, dass alle komplexen Nullstellen der Riemannschen Zeta-Funktion $\zeta(x)$, deren Realteil zwischen 0 und 1 liegen, einen Realteil von genau $\frac{1}{2}$ haben. Die verallgemeinerte Riemannsche Vermutung ist die entsprechende Annahme für bestimmte Verallgemeinerungen der Zeta-Funktion $\zeta(x)$, die sogenannten Dirichletschen L-Reihen. Details finden sich z.B. in [BS66].

Kapitel 2

Klassengruppen algebraischer Zahlkörper

In diesem Kapitel behandeln wir die mathematischen Grundlagen für unsere Arbeit. Wir geben eine kurze Einführung in Klassengruppen algebraischer Zahlkörper sowie deren Objekte. Wir zeigen, dass man prinzipiell mit diesen Objekten konstruktiv rechnen kann. Für eine ausführlichere Einführung in die Theorie der algebraischen Zahlkörper und deren Objekte und Algorithmen verweisen wir auf [BS66, Coh95].

2.1 Algebraische Zahlkörper

Ein *algebraischer Zahlkörper* ist ein $\mathbb{Q}$-Vektorraum endlicher Dimension, der zusätzlich ein Teilkörper von $\mathbb{C}$ ist. Die Dimension heißt *Grad des Zahlkörpers*. Sei K ein algebraischer Zahlkörper vom Grad n. Dann existiert ein Element $\varrho \in K$, das für K eine $\mathbb{Q}$-Basis der Form $\{1, \varrho, \varrho^2, \ldots, \varrho^{n-1}\}$ liefert. K ist der kleinste Teilkörper von $\mathbb{C}$, der alle rationalen Zahlen und ϱ enthält, in Zeichen $K = \mathbb{Q}(\varrho)$. Die Elemente aus K heißen *algebraische Zahlen*. Jedes Element $\alpha \in K$ besitzt eine eindeutige Darstellung der Form

$$\alpha = \sum_{j=0}^{n-1} q_j \varrho^j, \qquad q_j \in \mathbb{Q} \text{ für } 0 \leqslant j \leqslant n-1\,. \tag{2.1}$$

Das Element ϱ ist Nullstelle eines eindeutig bestimmten monischen (d.h. höchster Koeffizient ist 1) irreduziblen Polynoms $f(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_0 \in \mathbb{Q}[x] - \{0\}$. Dieses Polynom heißt *erzeugendes Polynom*. Die übrigen Nullstellen $\varrho^{(2)}, \ldots, \varrho^{(n)}$ von $f(x)$ heißen die *Konjugierten* von $\varrho =: \varrho^{(1)}$. Für ein Körperelement α mit der Darstellung (2.1) sind analog die *Konjugierten* von α definiert als $\alpha^{(i)} = \sum_{j=0}^{n-1} q_j \varrho^{(i)j}$ für $1 \leqslant i \leqslant n$.

Die Abbildungen, die $\varrho^{(1)} = \varrho$ auf eine Nullstelle $\varrho^{(i)}$ des Polynoms abbilden und die Elemente aus $\mathbb{Q}$ auf sich selbst abbilden, lassen sich jeweils zu homomorphen *Einbettungen* (Monomorphismen) σ_i von $\mathbb{Q}(\varrho)$ in $\mathbb{C}$ fortsetzen: $\sigma_i(\alpha) = \alpha^{(i)}$, $i \in \{1, \ldots, n\}$. Ist $\varrho^{(i)} \in \mathbb{R}$, so nennen wir σ_i eine *reelle Einbettung*, ansonsten eine *komplexe Einbettung*. Jeder algebraische Zahlkörper vom Grad n besitzt r reelle Einbettungen und $2s$ komplexe Einbettungen, wobei $r, s \in \{0, \ldots, n\}$

und $r+2s=n$ ist. Eine algebraische Zahl $\alpha \in K$ können wir auch als *(ausführlichen) Konjugiertenvektor* $\underline{\alpha} = \left(\sigma_1(\alpha), \ldots, \sigma_r(\alpha), \sigma_{r+1}(\alpha), \ldots, \sigma_{r+\frac{n-r}{2}}(\alpha), \overline{\sigma_{r+1}(\alpha)}, \overline{\sigma_{r+\frac{n-r}{2}}(\alpha)}\right)^T$ auffassen, wobei $\sigma_1, \ldots, \sigma_r$ reelle Einbettungen, $\sigma_{r+1}, \ldots, \sigma_{r+\frac{n-r}{2}}$ komplexe Einbettungen sind und mit $\overline{z}$ das Konjugiert-Komplexe von $z \in \mathbb{C}$ bezeichnet wird. Diese (redundante) Repräsentation können wir vereinfachen, indem wir s konjugiert-komplexe Komponenten weglassen. Wir erhalten den *(vereinfachten) Konjugiertenvektor*

$$\underline{\alpha} = \left(\sigma_1(\alpha), \ldots, \sigma_r(\alpha), \operatorname{Re}\sigma_{r+1}(\alpha), \operatorname{Im}\sigma_{r+1}(\alpha), \ldots, \operatorname{Re}\sigma_{r+\frac{n-r}{2}}(\alpha), \operatorname{Im}\sigma_{r+\frac{n-r}{2}}(\alpha)\right)^T \in \mathbb{R}^n \ . \tag{2.2}$$

Die *Norm von* α (in K) ist definiert als $N(\alpha) := \prod_{j=1}^{n} \sigma_j(\alpha)$. Für einen Zahlkörper K mit erzeugendem Polynom $f(x) \in \mathbb{Q}[x]$ mit Nullstelle ϱ gilt die Isomorphie

$$\begin{aligned} K = \mathbb{Q}(\varrho) \quad &\simeq \quad \mathbb{Q}[x]/(f(x)\cdot\mathbb{Q}[x]) \\ \alpha = \sum_{j=0}^{n-1} q_j \varrho^j \quad &\mapsto \quad \sum_{j=0}^{n-1} q_j x^j \mod f(x) \ . \end{aligned} \tag{2.3}$$

Die Elemente von K können somit als Restklassen $g+(f) := g + f\cdot\mathbb{Q}[x]$ mit $g \in \mathbb{Q}[x]$ aufgefasst werden. Eine beliebige solche Restklasse hat einen eindeutig bestimmten Vertreter vom Grad kleiner n. Dieser eindeutig bestimmte Vertreter entsteht aus einem beliebigen Vertreter durch Division mit Rest durch das erzeugende Polynom f. Wir nehmen nun an, die Körperelemente seien durch ihre eindeutig bestimmten Vertreter vom Grad kleiner n repräsentiert. Addition, Subtraktion und Multiplikation von Körperelementen können somit zurückgeführt werden auf die entsprechenden Operationen mit den entsprechenden Polynomen (Vertretern), wobei anschließend jeweils eine Division mit Rest durch das Polynom f zu erfolgen hat. Für ein Element $g+(f) \neq (f)$ ist $h+(f) \in \mathbb{Q}[x]/(f(x)\cdot\mathbb{Q}[x])$ das *Inverse*, wenn ein Polynom $z \in \mathbb{Q}[x]$ existiert mit $gh = 1 + zf$, d.h. $gh \equiv 1 \mod f$. Mit dem erweiterten Euklidischen Algorithmus können entsprechende Polynome h, z gefunden werden, da f irreduzibel ist. Die elementaren Grundrechenarten in K können somit in Polynomzeit durchgeführt werden.

2.2 Ordnungen

Ist eine algebraische Zahl α darstellbar als Nullstelle eines über $\mathbb{Q}[x]$ irreduziblen monischen Polynoms $g(x) \in \mathbb{Z}[x] - \{0\}$, so heißt α *ganz-algebraisch*.

Ein Ring $\mathcal{O}$ mit Eins mit $\mathbb{Z} \subseteq \mathcal{O} \subseteq K$, der zugleich freier $\mathbb{Z}$-Modul maximalen Rangs aus ganz-algebraischen Zahlen ist, heißt *Ordnung* des algebraischen Zahlkörpers K. Die bzgl. Inklusion eindeutig bestimmte maximale Ordnung heißt *Maximalordnung* von K, in Zeichen $\mathcal{O}_K$. Die Maximalordnung $\mathcal{O}_K$ ist gerade die Menge aller ganz-algebraischen Zahlen von K und wird daher auch *Ganzheitsring* genannt. Jedes Element $\alpha \in K$ kann als $\alpha = \beta\cdot\gamma^{-1}$ mit $\beta, \gamma \in \mathcal{O}_K$ geschrieben werden, analog zur Situation $K = \mathbb{Q}$ und $\mathcal{O}_{\mathbb{Q}} = \mathbb{Z}$. Der Ganzheitsring $\mathcal{O}_K$ kann von einer n-elementigen Menge $\{\omega_1 = 1, \omega_2, \ldots, \omega_n\} \subseteq \mathcal{O}$ ganz-algebraischer Zahlen erzeugt werden, die wir dann eine *Ganzheitsbasis* von K nennen. Die *Diskriminante einer Ordnung* $\mathcal{O}$ mit $\mathbb{Z}$-Basis $\{\omega_1, \omega_2, \ldots, \omega_n\}$ ist die Zahl

$$\Delta = \Delta_{\mathcal{O}} = \left[\det(\sigma_i(\omega_j))_{i,j}\right]^2, \qquad 1 \leqslant i, j \leqslant n.$$

Sie ist unabhängig von der Wahl der Basis. Die Diskriminante der Maximalordnung eines Zahlkörpers K heißt *Diskriminante des Zahlkörpers*, in Zeichen $\Delta = \Delta_K$. Ist in $K = \mathbb{Q}(\varrho)$ das Element ϱ ganz-algebraisch, so ist $\mathbb{Z}[\varrho]$ eine Ordnung; diese wird als *Gleichungsordnung* bezeichnet.

2.3 Einheiten und Regulator

Ein Element α einer Ordnung $\mathcal{O}$ heißt *Einheit*, falls $\alpha^{-1} \in \mathcal{O}$. In dem Falle gilt $|N(\alpha)| = 1$. Unterscheiden sich zwei Elemente einer Ordnung nur durch Multiplizieren mit einer Einheit, so heißen die beiden Elemente *assoziiert*. Die Menge aller Einheiten in $\mathcal{O}$ wird mit $\mathcal{O}^*$ bezeichnet. Die Menge ${\mathcal{O}_K}^*$ aller Einheiten der Maximalordnung bildet eine multiplikative Abelsche Gruppe, die *Einheitengruppe*. Ihre Elemente heißen auch *Einheiten von* K. Sei K ein algebraischer Zahlkörper vom Grad n mit r reellen und $2s$ komplexen Einbettungen in $\mathbb{C}$. Nach dem Dirichletchen Einheitensatz existieren eine n-te Einheitswurzel ξ und Einheiten $\eta_1, \ldots, \eta_{r+s-1}$ von K, so dass jede Einheit η von K eindeutig dargestellt werden kann in der Form

$$\eta = \xi^{z_0} \cdot \prod_{j=1}^{r+s-1} \eta_j^{z_j}, \qquad 0 \leqslant z_0 < n \text{ und } z_j \in \mathbb{Z} \ .$$

Die Zahl $r + s - 1$ heißt *Einheitenrang* von K. Die Menge $\{\eta_1, \ldots, \eta_{r+s-1}\}$ heißt System von *Fundamentaleinheiten* von K. Das Paar (r, s) heißt *Signatur* des Zahlkörpers.

Für eine beliebige Zahl $\alpha \in K$ betrachten wir den ausführlichen Konjugiertenvektor $\underline{\alpha}$ (siehe oben). Dann betrachten wir die Beträge $|\alpha|_j = |\sigma_j(\alpha)|$ für $1 \leqslant j \leqslant r$ und $|\alpha|_j = |\sigma_j(\alpha)|^2$ für $r + 1 \leqslant j \leqslant r + s$. Für ein System von Fundamentaleinheiten von K sei die Zahl R definiert als

$$R = |\det(\ln |\eta_i|_j)_{i,j}|, \qquad 1 \leqslant i, j \leqslant r - 1 \tag{2.4}$$

Die positive reelle Zahl R ist unabhängig von der konkreten Wahl des Systems der Fundamentaleinheiten; sie heißt *Regulator* von K.

2.4 Ideale

Sei $\mathcal{O}$ eine Ordnung. Ein *ganzes $\mathcal{O}$-Ideal (Ideal von $\mathcal{O}$)* ist eine additive Untergruppe $\mathfrak{a}$ von $\mathcal{O}$, die gleichzeitig ein $\mathcal{O}$-Modul ist, d.h. $\mathcal{O}\mathfrak{a} = \{\omega\alpha |\ \omega \in \mathcal{O}, \alpha \in \mathfrak{a}\} \subseteq \mathfrak{a}$. Die *Norm des Ideals* $\mathfrak{a}$ ist der Index von $\mathfrak{a}$ in $\mathcal{O}$, in Zeichen $N(\mathfrak{a})$. Ist $\mathcal{O}$ der Ganzheitsring, so sagt man auch, $\mathfrak{a}$ sei Ideal von K. Ist $\mathfrak{a} = \alpha\mathcal{O}$ für ein $\alpha \in K$, so heißt $\mathfrak{a}$ *Hauptideal*. Ist $\mathfrak{a}$ ein ganzes Ideal in $\mathcal{O}$, dann heißt jede Teilmenge der Form $\frac{1}{d}\mathfrak{a}$ mit $d \in \mathbb{Z} - \{0\}$ *gebrochenes $\mathcal{O}$-Ideal*. Die *Norm dieses gebrochenen Ideals* ist $N(\frac{1}{d}\mathfrak{a}) = \frac{N(\mathfrak{a})}{d^2}$.

Wir sagen, ein ganzes $\mathcal{O}$-Ideal $\mathfrak{a}$ *teile* ein ganzes $\mathcal{O}$-Ideal $\mathfrak{b}$, in Zeichen $\mathfrak{a} \mid \mathfrak{b}$, wenn $\mathfrak{b} \subseteq \mathfrak{a}$ gilt. Ein ganzes $\mathcal{O}$-Ideal $\mathfrak{p}$ heißt *Primideal*, falls für je zwei ganze $\mathcal{O}$-Ideale $\mathfrak{a}, \mathfrak{b}$ aus $\mathfrak{p} \mid \mathfrak{a}\mathfrak{b}$ stets $\mathfrak{p} \mid \mathfrak{a}$ oder $\mathfrak{p} \mid \mathfrak{b}$ folgt. Ist $\mathfrak{a}$ ein ganzes $\mathcal{O}$-Ideal, so existieren (bis auf die Reihenfolge) eindeutig bestimmte, paarweise verschiedene Primideale $\mathfrak{p}_1, \ldots, \mathfrak{p}_k$ mit

$$\mathfrak{a} = \prod_{i=1}^{k} \mathfrak{p}_i^{e_i} \ , \qquad e_i \in \mathbb{Z}_{>0},\ k \in \mathbb{Z}_{\geqslant 0} \ .$$

Für gebrochene $\mathcal{O}$-Ideale $\mathfrak{a}, \mathfrak{b}$ ist ihr *Idealprodukt* definiert als

$$\mathfrak{ab} = \left\{ \sum_{(\alpha,\beta)\in S} \alpha\beta \mid S \subset \mathfrak{a} \times \mathfrak{b} \text{ endlich} \right\} . \tag{2.5}$$

$\mathfrak{ab}$ ist wieder ein gebrochenes $\mathcal{O}$-Ideal. Der *Quotient* von $\mathfrak{a}$ und $\mathfrak{b}$ ist das gebrochene $\mathcal{O}$-Ideal $\mathfrak{a} : \mathfrak{b} = \{\alpha \in K \mid \alpha\mathfrak{b} \subseteq \mathfrak{a}\}$. Das gebrochene $\mathcal{O}$-Ideal $\mathfrak{a}$ wird als *invertierbar* bezeichnet, wenn $\mathfrak{a}(\mathcal{O} : \mathfrak{a}) = \mathcal{O}$. Die Menge der von $(0) = 0\mathcal{O}$ verschiedenen invertierbaren $\mathcal{O}$-Ideale bildet mit der Idealmultiplikation eine Abelsche Gruppe $\mathcal{I}(\mathcal{O})$. Die von (0) verschiedenen Hauptideale bilden hierin eine Untergruppe $\mathcal{H}(\mathcal{O})$. Die Faktorgruppe

$$Cl(\mathcal{O}) := \mathcal{I}(\mathcal{O})/\mathcal{H}(\mathcal{O}) \tag{2.6}$$

heißt *Klassengruppe von* $\mathcal{O}$. Zwei gebrochene Ideale $\mathfrak{a}$ und $\mathfrak{b}$ heißen *äquivalent*, wenn ein $\alpha \in K$ existiert mit $\mathfrak{b} = \alpha\mathfrak{a}$. Äquivalenz von Idealen ist eine Äquivalenzrelation, die mit der Idealmultiplikation verträglich ist. Die Äquivalenzklasse von $\mathfrak{a}$ heißt *Idealklasse* und wird mit $[\mathfrak{a}]$ bezeichnet. Eine Idealklasse wird durch ein beliebiges Ideal dieser Klasse eindeutig dargestellt. $Cl(\mathcal{O})$ ist eine endliche Abelsche Gruppe; ihre Ordnung heißt *Klassenzahl* $h(\mathcal{O})$. Ist $\mathcal{O} = \mathcal{O}_K$ der Ganzheitsring, so schreibt man statt $Cl(\mathcal{O}_K)$ auch $Cl(K) := \mathcal{I}(K)/\mathcal{H}(K)$ und spricht von der *Klassengruppe von* K; ihre Ordnung heißt *Klassenzahl von* K, in Zeichen $h = h(K)$. Die Klassenzahl einer Ordnung eines algebraischen Zahlkörpers ist im Allgemeinen unbekannt und kann auch nicht effizient berechnet werden (siehe Abschnitt 3.1).

2.5 Idealarithmetik

Ein gebrochenes $\mathcal{O}$-Ideal $\mathfrak{a}$ kann durch eine $\mathbb{Z}$-Basis $(\alpha_1, \ldots, \alpha_n)$ repräsentiert werden, wobei $\alpha_i \in K, 1 \leqslant i \leqslant n$. Wie wir bereits weiter oben erläutert haben, können diese Körperelemente durch Polynome vom Grad kleiner n dargestellt werden. Ein solches Polynom $g(x) = g_1x^{n-1} + \ldots + g_{n-1}x + g_n$ stellen wir mittels des Koeffizientenvektors $(g_1, \ldots g_n)$ dar. Das gebrochene Ideal $\mathfrak{a}$ kann dann als Gitter in $\mathbb{Q}^n$ (d.h. als eine diskrete additive Untergruppe von $\mathbb{Q}^n$) aufgefasst werden. Seien $\mathfrak{a}, \mathfrak{b}$ gebrochene $\mathcal{O}$-Ideale mit $\mathbb{Z}$-Basisdarstellungen $(\alpha_1, \ldots, \alpha_n)$ bzw. $(\beta_1, \ldots, \beta_n)$, dann besitzt das Idealprodukt $\mathfrak{ab}$ das Erzeugendensystem $(\alpha_i\beta_j)_{1\leqslant i,j\leqslant n}$. Dieses Erzeugendensystem kann wiederum als Gitter in $\mathbb{Q}^n$ aufgefasst werden. Durch Berechnung der Hermite-Normalform (siehe [Coh95]) kann in Polynomzeit eine Basis dieses Produktes bestimmt werden.

Wir erklären, wie man die „Größe“ (d.h. die Norm) von Idealen reduzieren kann, so dass eine effizientere Darstellung der betreffenden Idealklasse möglich ist. Sei $\mathfrak{a}$ ein gebrochenes $\mathcal{O}$-Ideal. $\mathfrak{a}$ heißt *c-reduziert* ($c \in \mathbb{R}_{>0}$), wenn das gebrochene $\mathcal{O}$-Ideal $\mathcal{O} : \mathfrak{a}$ das Element 1 enthält und

$$\{\alpha \in (\mathcal{O} : \mathfrak{a})| \ |\alpha^{(i)}| < 1/c \text{ für alle } i \in \{1, \ldots, n\}\} = \{0\}.$$

Wenn $\mathfrak{a}$ c-reduziert ist, so ist $\mathfrak{a}$ ein ganzes Ideal mit $N(\mathfrak{a}) \leqslant c^n\sqrt{|\Delta|}$. Das Bestimmen eines 1-reduzierten Ideals, welches zu vorgegebenem Ideal äquivalent ist, erfordert die Berechnung eines kürzesten Vektors in einem Gitter der Dimension n. Hierfür sind nur Algorithmen bekannt, die im Zahlkörpergrad n exponentielle Komplexität aufweisen. Daher verwendet man in der Praxis häufig den LLL-Reduktionsalgorithmus (siehe z.B. [Coh95]), welcher in Polynomzeit arbeitet, aber schlechtere Reduktionsergebnisse (2^n-reduzierte Ideale) liefert. In einer Idealklasse gibt es im Allgemeinen sehr viele verschiedene LLL-reduzierte (d.h. 2^n-reduzierte) Ideale. Hieraus ergibt sich

das nichttriviale Problem zu entscheiden, ob zwei vorgegebene Idealklassen gleich sind. Auf dieses Entscheidungsproblem werden wir in Kapitel 5 eingehen.

In Kapitel 5 werden wir auch erklären, wie man in Klassengruppen Elemente zufällig und gleichverteilt auswählt. Wir werden dort zudem effizientere Darstellungen und Algorithmen für die Objekte algebraischer Zahlkörper vorstellen.

Kapitel 3

NF-Kryptoverfahren: Entwurf, Sicherheit und Effizienz

In dieser Arbeit wollen wir nachweisen, dass kryptographische Verfahren über algebraischen Zahlkörpern *(NF-Kryptoverfahren)* eine Alternative zu gängigen Public-Key-Kryptoverfahren darstellen. Im Allgemeinen kennen wir aber nicht die Klassenzahl, d.h. die Ordnung der betreffenden Klassengruppen. Die meisten gängigen Kryptoverfahren setzen aber die Kenntnis der Ordnung der zugrunde liegenden Gruppe voraus. Bis jetzt war somit vollkommen unklar, welche kryptographischen Protokolle man in endlichen Abelschen Gruppen mit unbekannter Gruppenordnung (wie Klassengruppen algebraischer Zahlkörper) implementieren kann. In diesem Kapitel schließen wir diese Lücke. Wir zeigen, dass sich alle gängigen Typen kryptographischer Protokolle über endlichen Abelschen Gruppen mit unbekannter Gruppenordnung sicher und effizient[1] implementieren lassen. Hierzu entwerfen wir entsprechende kryptographische Verfahren und untersuchen deren Effizienz und Sicherheit.

Im Folgenden stützen wir uns teilweise auf unsere Arbeiten [BBHM02, BBHM99, BBHM00]. Über die Sicherheit und Effizienz einiger der von uns diskutierten Protokolle bei Implementierung in imaginär-quadratischen Zahlkörpern wird auch in [BH01, Ham02] berichtet. In den genannten Arbeiten werden lediglich Signaturverfahren mit gewöhnlicher Funktionalität und Verschlüsselungsverfahren diskutiert. Die Arbeiten geben keinen Aufschluss darüber, ob und wie man andere Typen kryptographischer Protokolle in Gruppen mit unbekannter Ordnung implementieren kann: spezielle Signaturverfahren wie Nichtabstreitbare Signaturverfahren (Undeniable Signature Schemes), Blinde Signaturverfahren, Gruppensignaturen, Schlüsselaustauschverfahren, Identifikationsverfahren, Faire Public-Key-Kryptosysteme, Pseudozufallszahlengeneratoren, Bit Commitment und fairer Münzwurf, Secret Sharing und Subliminal Channel.

Wir treten somit im vorliegenden Kapitel erstmals den Beweis an, dass sich in endlichen Abelschen Gruppen G mit unbekannter Gruppenordnung wie Klassengruppen algebraischer Zahlkörper alle gängigen Typen kryptographischer Protokolle effizient implementieren lassen, sofern sich folgende Operationen für beliebige Gruppenelemente $\alpha, \beta \in G$ effizient (d.h. Laufzeit polynomiell in der Eingabelänge) ausführen lassen:

1. die Gruppenoperation $\alpha \cdot \beta$

[1] Die Effizienz hängt natürlich von der Effizienz der Arithmetik in der verwendeten Gruppe ab.

2. Berechnung des Inversen α^{-1}
3. Gleichheitsentscheid $\alpha \stackrel{?}{=} \beta$
4. Auswahl eines zufälligen Elementes $\alpha \in G$ mit Gleichverteilung

In diesem Kapitel nehmen wir an, dass G eine endliche Abelsche Gruppe ist, in der sich die genannten Operationen effizient ausführen lassen. Zudem nehmen wir an, es gäbe keinen effizienten probabilistischen Algorithmus zur Berechnung eines großen Teilers der Gruppenordnung $|G|$ (oder gar zur Berechnung von $|G|$).

Dieses Kapitel gliedert sich wie folgt:
In Abschnitt 3.1 stellen wir die algorithmischen Probleme vor, auf denen die Sicherheit der von uns vorgestellten Kryptoverfahren basiert. Wir erläutern, in welchem Zusammenhang diese algorithmischen Probleme stehen.
In Abschnitt 3.2 modellieren wir den Begriff der Sicherheit. Wir erklären, was wir darunter verstehen, dass ein kryptographisches Protokoll „sicher“ sei. Wir gehen dabei explizit auf Signaturverfahren, Verschlüsselungsverfahren, Schlüsselaustauschverfahren, Identifikationsverfahren, Pseudozufallszahlengeneratoren und Bit Commitment-Verfahren ein.
In Abschnitt 3.3 stellen wir zunächst Strategien vor, kryptographische Protokolle zu konstruieren, die ohne Kenntnis der Gruppenordnung auskommen. Diese Strategien wenden wir beim anschließenden Entwurf von kryptographischen Protokollen an. Dabei stellen wir zunächst vor, welche kryptographische Protokolle sich bei unserer Literaturrecherche als leicht übertragbar auf endliche Abelsche Gruppen unbekannter Ordnung herausstellen. Zweitens präsentieren wir neue, in unserer Arbeitsgruppe entwickelte Kryptoverfahren, an deren Entwurf wir maßgeblich beteiligt waren. Schließlich entwickeln wir vollkommen neue kryptographische Protokolle. Wir untersuchen jeweils die Sicherheit und Effizienz der Protokolle.

In Kapitel 4 werden wir die effizientesten Algorithmen zum Lösen der zugrunde liegenden algorithmischen Probleme analysieren; wir werden sehen, dass die Probleme nach heutigem Kenntnisstand nicht effizient gelöst werden können, sofern unsere Empfehlungen zur Auswahl der Klassengruppen algebraischer Zahlkörper befolgt werden.

3.1 Die zugrunde liegenden algorithmischen Probleme

In diesem Abschnitt stellen wir die algorithmischen Probleme vor, auf denen die Sicherheit der von uns vorgestellten Kryptoverfahren basiert. Wir zeigen, in welchem Zusammenhang diese algorithmischen Probleme stehen.

3.1.1 Das diskrete Logarithmusproblem

Definition 3.1.1 (Diskretes Logarithmusproblem (DLP))
Seien eine endliche Abelsche Gruppe G und Gruppenelemente $\gamma, \alpha \in G$ gegeben. Das diskrete Logarithmusproblem $DLP(G, \alpha, \gamma)$ lautet:
Entscheide, ob $\alpha \in \langle\gamma\rangle$ ist. Falls $\alpha \in \langle\gamma\rangle$ ist, finde die kleinste positive ganze Zahl x, so dass $\gamma^x = \alpha$ ist.
x heißt diskreter Logarithmus *von α zur Basis γ. Wir schreiben: $x = \log_\gamma \alpha$.*

Diskrete Logarithmusprobleme in der Klassengruppe eines algebraischen Zahlkörpers bezeichnen wir mit NF-DLP.

3.1.2 Wurzelprobleme

Definition 3.1.2 (Spezielles Wurzelproblem (Root Problem) $\mathrm{RP}_{p\nmid|G|}$)
Seien eine endliche Abelsche Gruppe G, ein Gruppenelement $\alpha \in G$ und eine positive ganze Zahl n mit $\mathrm{ggT}(n, |G|) = 1$ gegeben. Das Wurzelproblem $\mathrm{RP}_{p\nmid|G|}(G, n, \alpha)$ lautet:
Berechne das Element $\xi \in G$, so dass $\xi^n = \alpha$ ist.
ξ heißt n-te Wurzel *von α in G. Das Problem, ξ zu bestimmen, nennen wir auch* spezielles n-tes Wurzelproblem.

Definition 3.1.3 (Allgemeines Wurzelproblem (Root Problem) $\mathrm{RP}_{p||G|}$)
Seien eine endliche Abelsche Gruppe G, ein Gruppenelement $\alpha \in G$ und eine Primzahl p mit $p \mid |G|$ gegeben. Das Wurzelproblem $\mathrm{RP}_{p||G|}(G, p, \alpha)$ lautet:
Entscheide, ob ein Element $\xi \in G$ existiert mit $\xi^p = \alpha$, und bestimme in diesem Fall ein solches $\xi \in G$.
ξ heißt p-te Wurzel *von α in G.*

Wurzelprobleme in der Klassengruppe eines algebraischen Zahlkörpers bezeichnen wir mit $\mathrm{NF} - \mathrm{RP}_{p\nmid|G|}$ bzw. mit $\mathrm{NF} - \mathrm{RP}_{p||G|}$.

3.1.3 Das Diffie-Hellman-Problem

Definition 3.1.4 (Diffie-Hellman Problem (DHP))
Seien eine endliche Abelsche Gruppe G und Gruppenelemente $\gamma, \gamma^a, \gamma^b \in G$ gegeben, wobei a, b ganze Zahlen sind. Das Diffie-Hellman-Problem $DHP(G, \gamma, \gamma^a, \gamma^b)$ lautet:
Berechne γ^{ab}.

Diffie-Hellman-Probleme in der Klassengruppe eines algebraischen Zahlkörpers bezeichnen wir mit NF-DHP.

3.1.4 Das Ordnungsproblem

Definition 3.1.5 (Ordnungsproblem (Order Problem, OP))
Seien eine endliche Abelsche Gruppe G und ein Gruppenelement $\gamma \in G$ gegeben. Das Ordnungsproblem $OP(G, \gamma)$ lautet:
Berechne ein von Null verschiedenes Vielfaches der Ordnung $\mathrm{ord}_G\, \gamma$ von γ, d.h. bestimme eine ganze Zahl $x \neq 0$, so dass $\gamma^x = 1_G$ ist.
Die kleinste positive Lösung x heißt Ordnung *von γ in G, wir schreiben dann $x = \mathrm{ord}_G\, \gamma$.*

Ordnungsprobleme in der Klassengruppe eines algebraischen Zahlkörpers bezeichnen wir mit NF-OP.

3.1.5 Das Gruppenordnungsproblem

Definition 3.1.6 (Gruppenordnungsproblem (Group order problem, GOP))
Sei eine endliche Abelsche Gruppe G gegeben. Das Gruppenordnungsproblem $GOP(G)$ lautet: Berechne die Gruppenordnung $|G|$.

Das Gruppenordnungsproblem in der Klassengruppe einer Ordnung der Diskriminante Δ eines algebraischen Zahlkörpers heißt *Klassenzahlproblem (Class number problem)*, in Zeichen NF $h(\Delta)$.

3.1.6 Das Faktorisierungsproblem

Definition 3.1.7 (Faktorisierungsproblem (Integer factoring problem, IFP))
Sei eine natürliche Zahl n gegeben. Das Faktorisierungsproblem $IFP(n)$ lautet: Entscheide, dass n eine Primzahl ist oder berechne ganze nichttriviale Faktoren n_1, n_2 von n mit $1 < n_1, n_2 < n$ und $n = n_1 n_2$.

3.1.7 Reduktionen und Schwierigkeit der algorithmischen Probleme

Wir zeigen nun die Beziehungen zwischen den im letzten Abschnitt vorgestellten Berechnungsproblemen NF-DLP, NF-OP, NF-RP und NF $h(\Delta)$ auf. Wir verwenden hierzu sog. Polynomzeitreduktionen zwischen Berechnungsproblemen. Den Begriff der Polynomzeitreduktionen führen wir unten ein. Zunächst erläutern wir den Begriff probabilistischer (Polynomzeit-)Algorithmus. Hierzu erklären wir zuerst probabilistische Polynomzeit-Turing-Maschinen. Eine Definition probabilistischer Polynomzeit-Turing-Maschinen findet sich beipielsweise in [GB01].

Informal ist eine *probabilistische Polynomzeit-Turing-Maschine* eine Turing-Maschine mit folgenden Eigenschaften:

1. Sie kann Münzen werfen, d.h. sie kann zufällige Bitfolgen erzeugen.
2. Ihr Verhalten kann von den Ergebnissen der Münzwürfe abhängen.
3. Ihre Laufzeit ist polynomiell in der Länge der Eingabe beschränkt.
4. Sie gibt nur zu einer gewissen Wahrscheinlichkeit ein korrektes Ergebnis aus.

Von jetzt an seien *effizienter Algorithmus* und *probabilistischer Polynomzeit-Algorithmus* Synonyme für eine probabilistische Polynomzeit-Turing-Maschine.

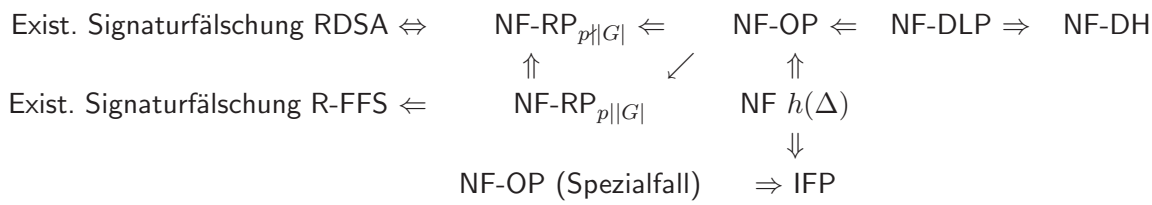
Nun erklären wir informal den Begriff *Polynomzeitreduktion von Berechnungsproblemen*. Eine exaktere Darstellung findet sich in [Ham02]. Für zwei Berechnungsprobleme A und B bedeutet

$A \Leftarrow B$**:** Unter der Annahme, es existiere ein Algorithmus mit konstantem Aufwand für Problem B, lässt sich nachweisen, dass dann auch ein probabilistischer Polynomzeit-Algorithmus für Problem A existiert. Informal gesprochen: Jeder probabilistische Lösungsalgorithmus für Problem B lässt sich zu einem probabilistischen Algorithmus für Problem A so modifizieren, dass der Algorithmus für A nur einen polynomiellen Mehraufwand gegenüber dem Algorithmus für B besitzt. Ein effizienter probabilistischer Algorithmus für B kann also zu einem effizienten probabilistischen Algorithmus für A ausgebaut werden. Existiert umgekehrt kein

effizienter probabilistischer Algorithmus für A, so kann es auch keinen effizienten probabilistischen Algorithmus für B geben. Man sagt, das Problem A lasse sich auf B *polynomiell reduzieren*. Informal gesprochen ist das Problem A *höchstens so schwer* wie B. Das Problem A kann als Spezialfall von B aufgefasst werden. Alternativ schreiben wir auch $B \Rightarrow A$.

$A \Leftrightarrow B$**:** Es gilt $A \Leftarrow B$ und $B \Leftarrow A$. Man sagt, die Berechnungsprobleme A und B seien *polynomiell äquivalent*. Gibt es also einen effizienten (d.h. polynomiellen) probabilistischen Algorithmus, der A löst, so lässt sich daraus auch ein effizienter (polynomieller) probabilistischer Algorithmus für B generieren und umgekehrt. Informal gesprochen sind die Probleme A und B *gleich schwer*.

Das folgende Schaubild gibt einen Überblick über die Zusammenhänge zwischen den Problemen in Klassengruppen, die den Bedingungen aus Abschnitt 4.2 genügen.[2]



Wir beweisen nun die oben schematisch dargestellten Reduktionen zwischen Berechnungsproblemen. Das Faktorisierungsproblem großer Zahlen (integer factoring problem, IFP) lässt sich im Allgemeinen nicht auf das Gruppenordnungsproblem (GOP) einer endlichen Abelschen Gruppe reduzieren. Ansonsten gelten alle oben dargestellten Reduktionen für beliebige endliche Abelsche Gruppen G, in denen effizient (d.h. in Polynomzeit) die Gruppenoperation ausgeführt werden kann. Für Reduktionen, an denen das Berechnungsproblem $\mathrm{RP}_{p||G|}$ beteiligt ist, setzen wir zudem voraus, dass für die betrachtete Primzahl p der p-Anteil[3] S_p der Gruppe relativ klein ist: $|S_p| \ll |G|$.[4] Genauer meinen wir hiermit: $|S_p|$ ist so klein, dass ein probabilistischer Polynomzeit-Algorithmus existiert, der $|S_p|$ Mal ein Gruppenelement aus S_p zufällig auswählt. Sei G fortan eine entsprechende endliche Abelsche Gruppe.

Proposition 3.1.8
$\mathrm{RP}_{p\nmid|G|} \Leftarrow \mathrm{RP}_{p||G|}$*: Das Wurzelproblem* $\mathrm{RP}_{p\nmid|G|}$ *lässt sich polynomiell reduzieren auf das Wurzelproblem* $\mathrm{RP}_{p||G|}$.

Beweis: Gegeben sei ein Gruppenelement $\upsilon \in G$ und eine Primzahl p mit $p \nmid |G|$. Wir suchen ein $\xi \in G$ mit $\xi^p = \upsilon$. Hierzu betten wir G isomorph ein in $G' := G \times \mathbb{Z}/p\mathbb{Z}$ vermöge $\varphi : G \xrightarrow{\simeq} G'$, $g \mapsto (\gamma, \alpha_2)$, wobei α_2 beliebig, aber fest aus $\mathbb{Z}/p\mathbb{Z}$ gewählt sei und für $\gamma_1, \gamma_2 \in G$ gelte: $\varphi(\gamma_1\gamma_2) = (\gamma_1, \alpha_2)(\gamma_2, \alpha_2) := (\gamma_1\gamma_2, \alpha_1 + \alpha_2)$. Nach Voraussetzung können wir eine p-te Wurzel $(\beta, \alpha) \in G'$ (mit $\beta \in G, \alpha \in \mathbb{Z}/p\mathbb{Z}$) von $\varphi(\upsilon) = (\upsilon, \alpha_2) \in G'$ berechnen: $(\beta, \alpha)^p = (\upsilon, \alpha_2)$. Hier gilt $\beta^p = \upsilon$ und $p\alpha = \alpha_2$. Also ist $\xi := \beta \in G$ p-te Wurzel von $\upsilon \in G$. □

[2] Diese Klassengruppen besitzen für kleine Primzahlen p einen kleinen p-Anteil (siehe unten).

[3] Für eine endliche Abelsche Gruppe G mit Primteiler p der Gruppenordnung ($p \mid |G|$) sei S_p die Menge aller Elemente in G, deren Ordnung eine Potenz von p ist (einschließlich des neutralen Elementes mit Ordnung $p^0 = 1$). S_p ist eine Untergruppe von G mit Ordnung p^k, $k \in \mathbb{Z}_{\geqslant 0}$. S_p heißt *p-Sylow-Gruppe* oder *p-Anteil von* G.

[4] In den von uns in Abschnitt 4.2 vorgeschlagenen Klassengruppen ist der p-Anteil für kleine Primteiler p der Klassenzahl klein. Dies folgt aus der Existenz großer Primteiler der Klassenzahl. Unsere Experimente zeigen, dass in der Praxis für die Größe des p-Anteils S_p gilt: $|S_p| = p^k$ mit $k \in \{1, 2, 3\}$.

Proposition 3.1.9
$\mathrm{RP}_{p\nmid|G|} \Leftarrow OP$: *Das Wurzelproblem* $\mathrm{RP}_{p\nmid|G|}$ *lässt sich polynomiell reduzieren auf das Ordnungsproblem OP.*

Beweis: Der Beweis findet sich beispielsweise in unserer Arbeit [BBHM02, Theorem 3]. □

Proposition 3.1.10
$\mathrm{RP}_{p||G|} \Leftarrow OP$: *Das Wurzelproblem* $\mathrm{RP}_{p||G|}$ *lässt sich polynomiell reduzieren auf das Ordnungsproblem OP.*

Beweis: Sei p Primteiler der Gruppenordnung $|G|$. Wir nehmen an (siehe oben), dass der p-Anteil von G klein ist: $|S_p| \ll |G|$. Zudem sei $\beta \in G$ gegeben. Wir suchen eine Lösung des Problems $\mathrm{RP}_{p||G|}(G, p, \beta)$, also ein $\alpha \in G$ mit $\alpha^p = \beta$. O.B.d.A gehen wir davon aus, dass wir effizient Ordnungen beliebiger Elemente von G bestimmen können. Entsprechend sei uns $\mathrm{ord}\,\alpha = p^r \cdot t$ mit $p \nmid t$, $t \in \mathbb{Z}_{\geqslant 1}$, $r \geqslant 0$ bekannt. Den (einfachen) Fall $r = 0$ haben wir bereits im Beweis zu Proposition 3.1.9 behandelt. Sei jetzt also $r \geqslant 1$. Wir zerlegen nun β in p-Anteil und Nicht-p-Anteil ($\beta = \beta_1\beta_2$ mit $\beta_1 \in S_p, \beta_2 \in U$), berechnen p-te Wurzeln im p-Anteil und Nicht-p-Anteil und setzen die Teillösungen zu einer Gesamtlösung zusammen. Im Detail gehen wir wie folgt vor: Wegen $\mathrm{ggT}(t, p^r) = 1$ können wir mit dem erweiterten Euklidischen Algorithmus ganze Zahlen x_1, x_2 berechnen mit $tx_1 + p^r x_2 = 1$. Berechne $\beta_1 := \beta^{tx_1} \in S_p$ und $\beta_2 := \beta^{p^r x_2} \in U$. Dann gilt $\beta_1\beta_2 = \beta^{tx_1 + p^r x_2} = \beta^1$. Bestimme eine p-te Wurzel $\alpha_1 \in S_p$ aus β_1; hierzu wählen wir ein beliebiges Element $\gamma \in G$, setzen $\alpha_1 := \gamma^t$ und testen, ob $\alpha_1^p = \beta_1$. Ist $\alpha_1^p \neq \beta_1$, wiederholen wir die Prozedur mit einem erneut zufällig gewählten $\gamma \in G$. Wir raten also eine Lösung $\alpha_1 \in S_p$. Da $|S_p|$ nach Voraussetzung klein ist, führt diese Strategie effizient zum Erfolg. Bestimme nun eine p-te Wurzel $\alpha_2 \in U$ aus β_2 gemäß der im Beweis zu Proposition 3.1.9 geschilderten Vorgehensweise. Setze $\alpha := \alpha_1\alpha_2$. Dann gilt $\alpha^p = \alpha_1^p\alpha_2^p = \beta_1\beta_2 = \beta$. □

Proposition 3.1.11
$OP \Leftarrow DLP$: *Das Ordnungsproblem OP lässt sich polynomiell reduzieren auf das diskrete Logarithmusproblem DLP.*

Beweis: Der Beweis findet sich beispielsweise in unserer Arbeit [BBHM02, Theorem 3]. □

Offensichtlich kann man durch Berechnen eines diskreten Logarithmus das Diffie-Hellman-Problem lösen:

Proposition 3.1.12
$DHP \Leftarrow DLP$: *Das Diffie-Hellman-Problem DHP lässt sich polynomiell reduzieren auf das diskrete Logarithmusproblem DLP.*

Die Ordnung einer Gruppe ist bereits Vielfaches der Ordnung eines beliebigen Elementes der Gruppe:

Proposition 3.1.13
$OP \Leftarrow GOP$: *Das Ordnungsproblem (für Gruppenelemente) lässt sich polynomiell reduzieren auf das Gruppenordnungsproblem.*

Proposition 3.1.14
IFP $\Leftarrow$ *NF-OP (Spezialfall): Das Faktorisierungsproblem lässt sich polynomiell reduzieren auf das Ordnungsproblem NF-OP in Klassengruppen imaginär-quadratischer Zahlkörper.*

Beweis: (Skizze) Ein Beweis findet sich beispielsweise in [Sch82, Mey97]. □

Das Ordnungsproblem in Klassengruppen imaginär-quadratischer Zahlkörper ist also mindestens so schwierig wie das Faktorisierungsproblem natürlicher Zahlen.

Gerade haben wir gezeigt, dass NF-RP $\Leftarrow$ NF-OP $\Leftarrow$ NF-DLP und NF-DHP $\Leftarrow$ NF-DLP gilt. Es ist unbekannt, ob NF-RP und NF-OP gleich schwierige Probleme sind (d.h. existiert für das eine Problem ein probabilistischer Polynomzeitalgorithmus, so auch für das andere) oder ob NF-OP echt schwieriger als NF-RP ist (d.h. es existiert zwar ein probabilistischer Polynomzeitalgorithmus für NF-RP, nicht aber für NF-OP). Entsprechendes gilt beispielsweise für die Beziehung zwischen NF-OP und NF-DLP: Man weiß nicht, ob die Probleme gleich schwer sind.

Nach heutigem Kenntnisstand sind die Probleme in der Praxis gleich schwierig: Das effizienteste Verfahren zum Lösen von NF-RP erfordert das Lösen des NF-OP. Das Lösen des NF-OP erfordert in der Praxis nach heutigem Kenntnisstand den gleichen Aufwand wie das Lösen des NF $h(\Delta)$-Problems, denn der gleiche probabilistische Algorithmus (siehe Kapitel 4) gilt heute für beide Probleme als am Effizientesten. Die schnellsten Algorithmen zum Lösen des NF $h(\Delta)$-Problems und des NF-DLP besitzen die gleiche Komplexität. Schließlich ist kein besseres Verfahren bekannt, um das NF-DHP zu lösen, als vorher ein NF-DLP zu lösen.

Verfahren zur Lösung des NF-DLP werden in Abschnitt 4.1 vorgestellt. Das beste bekannte Verfahren zur Lösung des NF-DLP ist für große Diskriminanten nicht effizient, und somit sind die besten bekannten Verfahren zur Lösung des NF-OP, des NF-RP und des NF-DHP für große Diskriminanten ebenfalls ineffizient.

Weiterhin sei bemerkt, dass kein effizientes Verfahren zur Berechnung der Klassenzahl, eines Vielfaches der Klassenzahl oder eines Teilers der Klassenzahl bekannt ist.

Im Abschnitt 4.2 geben wir notwendige Bedingungen für endliche Abelsche Gruppen G an, damit die aufgeführten Berechnungsprobleme in G schwer sind. Wir konstruieren in Abschnitt 4.3 Familien von algebraischen Zahlkörpern, deren Klassengruppen diese Bedingungen erfüllen und in denen die Berechnungsprobleme nach heutigem Kenntnisstand schwer sind.

3.2 Das Sicherheitsmodell

In diesem Abschnitt erklären wir, was wir darunter verstehen, dass ein kryptographisches Protokoll „sicher“ sei. Wir erläutern den Begriff „Sicherheit“ für Signaturverfahren (Abschnitt 3.2.1), Verschlüsselungsverfahren (Abschnitt 3.2.2), Schlüsselaustauschverfahren (Abschnitt 3.2.4), Identifikationsverfahren (Abschnitt 3.2.5), Pseudozufallszahlengeneratoren (Abschnitt 3.2.6) und Bit Commitment-Verfahren (Abschnitt 3.2.7). Kryptographische Hashfunktionen modellieren wir durch Zufallsorakel (siehe Abschnitt 3.2.3). Eine fundierte Übersicht über die Modellierung der Sicherheit kryptographischer Protokolle findet man in [GB01] oder [Gol99].

Wir gehen in diesem Kapitel von folgendem Szenario aus: **A** ist ein legitimer Signierer oder Empfänger von verschlüsselten Texten oder ein legitimer Beweiser. **B** ist **A**s legitimer Kommunikationspartner. Ein „legitimer“ Teilnehmer eines Protokolls folgt exakt den Protokollvorschriften. **E** ist ein Angreifer, der Zugriff auf **A**s öffentlichen Schlüssel, nicht aber auf **A**s privaten Schlüssel hat.

Wir modellieren **A**, **B** und **E** als probabilistische Polynomzeit-Turing-Maschinen (siehe Abschnitt 3.1.7). Das bedeutet insbesondere, die Angriffe von **E** auf die kryptographischen Protokolle müssen nicht deterministisch sein; sie besitzen eine gewisse Erfolgswahrscheinlichkeit.

Informal stufen wir ein kryptographisches Protokoll als sicher ein, wenn die Erfolgswahrscheinlichkeit für einen Angriff *vernachlässigbar* ist. In den folgenden Unterabschnitten erklären wir zunächst, wann ein Angriff erfolgreich ist, welche Ziele ein Angreifer also verfolgt. Zuerst definieren wir noch den Begriff „vernachlässigbar".

Definition 3.2.1
Eine Funktion $f : \mathbb{R} \to \mathbb{R}$ *heißt* vernachlässigbar, *wenn für jedes* $c \in \mathbb{R}_{>0}$ *gilt: Es existiert ein* $x_c \in \mathbb{R}_{>0}$, *so dass* $x^{-c} > |f(x)|$ *für alle* $x \in \mathbb{R}$ *mit* $x \geqslant x_c$.

3.2.1 Signaturverfahren

Wir definieren in diesem Abschnitt, wann wir ein Signaturverfahren sicher nennen. Eine Signatur von **A** für einen Text M bezeichnen wir mit $Sig_{\mathbf{A},M}$.

Angriffsziele
Ein Angreifer **E** kann die folgenden drei Angriffsziele gegen ein Signaturverfahren verfolgen (siehe [GB01]):

1. *Existenzielle Fälschung (existential forgery)*: **E** erzeugt eine gültige Signatur $Sig_{\mathbf{A},M}$ für mindestens einen Text M. **E** hat dabei keinen Einfluss auf die Wahl des Textes, für den die gültige Signatur berechnet werden kann.

2. *Universelle Fälschung (universal forgery)*: **E** erzeugt eine gültige Signatur $Sig_{\mathbf{A},M}$ für einen beliebigen Text M seiner Wahl.

3. *Schlüsselgewinn*: **E** berechnet **A**s privaten Schlüssel.

Das am schwierigsten zu erreichende Ziel ist ein Schlüsselgewinn, das am leichtesten zu erreichende Ziel ist eine existenzielle Fälschung. Ist für ein Signaturverfahren keine existenzielle Fälschung möglich, so ist erst recht keine universelle Fälschung oder ein Schlüsselgewinn möglich.

Angriffsmethoden
Bei einem Angriff auf ein Signaturverfahren berücksichtigen wir außerdem, welche Möglichkeiten einem Angreifer zur Verfügung stehen. Wir unterscheiden die folgenden Fälle (siehe [GB01]):

1. Angriff nur mit Kenntnis des öffentlichen Schlüssels des legitimierten Signierers *(Kein-Text-Angriff, no-message attack, key-only attack)*.

2. Angriff mit bekannten Text-Signatur-Paaren *(Bekannter-Text-Angriff, known-message attack)*. Der Angreifer hat die Texte, zu denen ihm gültige Signaturen zur Verfügung stehen, allerdings nicht auswählen können.

3. Angriff mit ausgewählten Texten *(Gewählter-Text-Angriff, chosen-message attack)*. Der Angreifer kann die Texte auswählen, zu denen ihm anschließend gültige Signaturen zur Verfügung stehen. Dabei wird noch feiner unterschieden, wann der Angreifer die Auswahl der Texte vornehmen kann, nämlich

(a) a priori, also vor dem eigentlichen Angriff. Der Angreifer erhält also gültige Signaturen für eine von ihm gewählte Liste von Texten, bevor er den eigentlichen Angriff startet. *(Nicht-angepasst-gewählter-Text-Angriff, non-adaptive-chosen-message attack)*

(b) während des Angriffs, wobei die Auswahl der Texte von bisher gewonnenen Ergebnissen (d.h. Signaturen) abhängt *(Angepasst-Gewählter-Text-Angriff, adaptive-chosen-message attack)*.

Der stärkste Angriff ist ein Angriff mit angepasst gewähltem Text. Ein Signaturverfahren, welches einem solchen Angriff widersteht, widersteht auch den Angriffen mit nicht-angepasst gewähltem Text, mit bekanntem Text und den Kein-Text-Angriffen.

Sicherheit von Signaturverfahren

Ein Signaturverfahren heißt *sicher*, wenn die Wahrscheinlichkeit vernachlässigbar klein ist, dass ein Angreifer **E** durch einen Angriff mit angepasst gewählten Texten (adaptive-chosen-message attack) die existenzielle Fälschung einer Signatur $Sig_{\mathbf{A},M}$ in polynomiell beschränkter Zeit erzeugen kann (siehe [GB01]).

3.2.2 Verschlüsselungsverfahren

Wir definieren in diesem Abschnitt, wann wir ein Public-Key-Verschlüsselungsverfahren sicher nennen. Einen Schlüsseltext, der mit dem privaten Schlüssel von **A** zum Klartext M entschlüsselt werden kann, bezeichnen wir mit $\mathcal{C}_{\mathbf{A},M}$.

Angriffsziele

Ein Angreifer **E** kann folgende Angriffsziele gegen ein Verschlüsselungsverfahren verfolgen:

1. *Unterscheidung*: **E** entscheidet für zwei vorgegebene Klartexte M_1 und M_2 und für vorgegebenen Schlüsseltext $\mathcal{C} \in \{\mathcal{C}_{\mathbf{A},M_1}, \mathcal{C}_{\mathbf{A},M_2}\}$ zu einem dieser Klartexte, ob $\mathcal{C} = \mathcal{C}_{\mathbf{A},M_1}$ oder $\mathcal{C} = \mathcal{C}_{\mathbf{A},M_2}$ ist.

2. *Entschlüsselung*: **E** entschlüsselt $\mathcal{C}_{\mathbf{A},M}$.

3. *Schlüsselgewinn*: **E** berechnet **A**s privaten Schlüssel.

Das am schwierigsten zu erreichende Ziel ist ein Schlüsselgewinn, das am leichtesten zu erreichende Ziel ist Unterscheidung. Ist für ein Verschlüsselungsverfahren für einen Angreifer keine Unterscheidung möglich, so ist erst recht keine Entschlüsselung oder ein Schlüsselgewinn möglich.

Angriffsmethoden

Bei einem Angriff auf ein Verschlüsselungsverfahren berücksichtigen wir außerdem, welche Möglichkeiten einem Angreifer zur Verfügung stehen. Wir unterscheiden die folgenden Fälle [MOV97, Kap. 1.13]:

1. Angriff nur mit Kenntnis des öffentlichen Schlüssels des legitimierten Empfängers einer Nachricht (eines Klartextes) *(Kein-Text-Angriff, no-message attack, key-only attack)*.

2. Angriff mit bekannten Klartext-Schlüsseltext-Paaren *(Bekannter-Klartext-Angriff, Known-Plaintext-Attack)*. Der Angreifer hat die Klartexte, zu denen ihm gültige Schlüsseltexte zur Verfügung stehen, allerdings nicht auswählen können.

3. Angriff mit ausgewählten Schlüsseltexten *(Gewählter-Schlüsseltext-Angriff, Chosen-Ciphertext-Attack)*. Der Angreifer kann die Schlüsseltexte auswählen, zu denen ihm anschließend die zugehörigen Klartexte zur Verfügung stehen. Dabei wird noch feiner unterschieden, wann der Angreifer die Auswahl der Schlüsseltexte vornehmen kann, nämlich

   (a) a priori, also vor dem eigentlichen Angriff. Der Angreifer erhält also die Klartexte, die zu einer von ihm gewählten Liste von Schlüsseltexten passen, bevor er den eigentlichen Angriff startet. *(Nicht-angepasst-gewählter-Schlüsseltext-Angriff, non-adaptive-chosen-ciphertext attack)*

   (b) während des Angriffs, wobei die Auswahl der Schlüsseltexte von bisher gewonnenen Ergebnissen (d.h. Klartexten) abhängt *(Angepasst-Gewählter-Schlüsseltext-Angriff, adaptive-chosen-ciphertext attack)*.

Der stärkste Angriff ist ein Angriff mit angepasst gewähltem Schlüsseltext. Ein Verschlüsselungsverfahren, welches einem solchen Angriff widersteht, widersteht auch den Angriffen mit nicht-angepasst gewähltem Schlüsseltext, mit bekanntem Schlüsseltext und den Kein-Text-Angriffen.

Sicherheit von Verschlüsselungsverfahren
Ein Verschlüsselungsverfahren heißt *sicher*, wenn die Wahrscheinlichkeit vernachlässigbar klein ist, dass ein Angreifer **E** durch einen Angriff mit angepasst gewählten Schlüsseltexten (adaptive-chosen-ciphertext attack) zwei Klartexte M_1 und M_2 in polynomiell beschränkter Zeit findet, die er in polynomiell beschränkter Zeit bezüglich $\mathcal{C}$ wie oben beschrieben unterscheiden kann (siehe [NY90], [Sho98]).

3.2.3 Das Random Oracle Model (Zufallsorakel)

Bis zum Jahr 1998 gelang für kein praktikables Signatur- oder Verschlüsselungsverfahren der Beweis der Sicherheit im Sinne der Abschnitte 3.2.1 und 3.2.2, sofern alleine angenommen wurde, dass ein zugrunde liegendes mathematisches Berechnungsproblem in der Praxis nicht gelöst werden kann. Bellare und Rogaway [BR93] modellierten daher für die Sicherheitsbeweise kryptographische Hashfunktionen durch *Zufallsorakel (Random Oracle)*. Das hieraus resultierende Modell heißt *Random Oracle Model*. Eine detaillierte Diskussion des Random Oracle Models findet sich in [BR93, CGH98]. Wir erklären das Konzept informal. Seien n und m fest gewählte, positive ganze Zahlen, und sei $f : \{0,1\}^m \to \{0,1\}^n$ eine Funktion, dann ist in unserem Kontext ein *Orakel für f* ein Mechanismus, der zu einer Eingabe $x \in \{0,1\}^m$ den Wert von $f(x)$ berechnet. Alles andere über f bleibt verborgen (daher die Bezeichnung „Orakel“). Wird f unter allen möglichen Abbildungen von $\{0,1\}^m$ nach $\{0,1\}^n$ zufällig ausgewählt, dann bezeichnet man das Orakel für f als *Random Oracle (Zufallsorakel)*. In den Beweisen über die Sicherheit kryptographischer Protokolle ersetzt man die Hashfunktion $h : \{0,1\}^m \to \{0,1\}^n$ üblicherweise durch ein Random Oracle. In der konkreten Implementierung des kryptographischen Protokolls verwendet man hingegen eine unter den Funktionen $\{0,1\}^m \to \{0,1\}^n$ gewählte Hashfunktion. Mit diesem Trick gelangen für viele kryptographische Protokolle Sicherheitsbeweise, welche allerdings nur im Random Oracle Model gültig sind.

Das Random Oracle Model ist in jüngster Vergangenheit auf Kritik gestoßen. Denn Canetti, Goldreich und Halevi [CGH98] zeigten, dass es Signaturverfahren und Verschlüsselungsverfahren gibt, die zwar sicher im Random Oracle Model sind, für die aber *jede praktische Implementierung* des Random Oracle Model als Hashfunktion zu einem praktisch *unsicheren Verfahren* führt. Die Autoren führen den Existenzbeweis an theoretisch konstruierten und gewissermaßen unnatürlichen Signatur- und Verschlüsselungsverfahren. Aus der Literatur sind viele Signatur- und Verschlüsselungsverfahren bekannt, deren Sicherheit im Random Oracle Model nachgewiesen werden konnte. Bis heute gibt es aber keinen Hinweis darauf, dass eines dieser Verfahren in der Praxis unsicher sei ([CGH98]).

Dennoch ist das ehemals hohe Ansehen des Random Oracle Model unter den Kryptologen nun beeinträchtigt. Mittlerweile wurden erste Verfahren vorgeschlagen, die nur spezielle Eigenschaften des Zufallsorakels benötigen, welche durch reale Implementierungen von Hashfunktionen abgebildet werden können ([Can97, CMR98, GHR99]). Cramer und Shoup schlugen in [CS98] ein Public-Key-Kryptosystem vor, das in der Praxis effizient und beweisbar sicher gegen adaptive chosen ciphertext attacks ist, sofern das Diffie-Hellman-Entscheidungsproblem (siehe z.B. [CS98]) in der Praxis nicht gelöst werden kann. Bei ihrem Sicherheitsbeweis kamen Cramer und Shoup ohne das Random Oracle Model aus. Damit gelang erstmals der Entwurf eines Public-Key-Verfahrens, welches effizient und - allein unter der Annahme der Schwierigkeit eines algorithmischen Problems - beweisbar sicher ist.

Führende Kryptologen sind sich einig darüber (siehe z.B. [CGH98] oder [CS98]), dass beweisbare Sicherheit eines kryptographischen Protokolls im Random Oracle Model zwar nicht zu beweisbarer Sicherheit praktischer Implementierungen der Protokolle führt, aber dennoch folgende Vorteile besitzt:

1. Im Random Oracle Model beweisbar sichere Verfahren halten bis heute in der Praxis allen bekannten Angriffen stand.

2. Das Random Oracle Model schließt eine große Klasse an Designfehlern in Protokollen aus, kann somit als Engineering-Tool und Test-Umgebung für kryptographische Protokolle dienen. Ein Protokoll, welches in dieser Test-Umgebung Schwächen offenbart, sollte verworfen werden.

3. Unter Benutzung des Random Oracle Model kann man einfache und effiziente Protokolle entwerfen, über die sich gewisse Sicherheitsaussagen beweisen lassen. In gewissem Sinne nachweisbare Sicherheit ist besser, als überhaupt keine Sicherheitsaussagen beweisen zu können.

Aus den genannten Gründen verwenden wir in der vorliegenden Arbeit das Random Oracle Model.

3.2.4 Schlüsselaustauschverfahren

Wir definieren in diesem Abschnitt, wann wir ein Schlüsselvereinbarungsverfahren sicher nennen.

Wir unterscheiden folgende Typen von Schlüsselvereinbarungsprotokollen[5]:

[5] Als Synonym für Schlüsselvereinbarungsprotokoll verwenden wir den Begriff Schlüsselaustauschprotokoll.

1. (Einfaches) Schlüsselvereinbarungsprotokoll *(key establishment protocol)*
   Ziel: Zwei Parteien einigen sich über einen unsicheren Kanal auf ein gemeinsames Geheimnis (shared secret).

2. Authentisches Schlüsselvereinbarungsprotokoll *(authenticated key establishment protocol)*
   Ziel: Zwei Parteien einigen sich über einen unsicheren Kanal auf ein gemeinsames Geheimnis (shared secret), wobei der Kommunikationspartner authentifiziert wurde. Wird nur eine Partei authentifiziert, sprechen wir von einem einseitigen *(unilateral)* authentischen Schlüsselvereinbarungsprotokoll. Verläuft die Authentifikation in beiden Richtungen, sprechen wir von einem beidseitigen *(mutual)* authentischen Schlüsselvereinbarungsprotokoll.

Angriffsziele

Gewinn des shared secret. Der Angreifer berechnet das zwischen zwei Teilnehmern vereinbarte shared secret.

Angriffsmethoden

Bei der Beurteilung der Sicherheit eines Verfahrens unterscheiden wir zwischen passiven und aktiven Angriffen. Bei einem *passiven Angriff* hat der Angreifer nur lesenden Zugriff auf den Kommunikationskanal. Hingegen hat ein Angreifer bei einem *aktiven Angriff* darüber hinaus die Möglichkeit, Nachrichten im Kanal abzufangen, zu modifizieren oder zusätzliche Nachrichten einzufügen.

Sicherheit von Schlüsselaustauschverfahren

Informal betrachten wir ein Schlüsselvereinbarungsprotokoll als *sicher* gegen passive oder aktive Angriffe, wenn kein entsprechender Angreifer in polynomiell beschränkter Zeit mit nichtvernachlässigbarer Wahrscheinlichkeit ein shared secret in Erfahrung bringen kann, welches zwischen zwei legitimen Teilnehmern (siehe oben) vereinbart wurde.

Eine akzeptierte formale Definition der Sicherheit von Schlüsselvereinbarungsprotokollen ist uns aus der Literatur nicht bekannt.

3.2.5 Identifikationsverfahren

Wir definieren in diesem Abschnitt, wann wir ein Identifikationsverfahren sicher nennen. Das von uns verwendete Sicherheitsmodell geht auf die in der Kryptographie weit akzeptierte Arbeit [FFS88] zurück. Für die exakten Definitionen der von uns informal eingeführten Begriffe verweisen wir auf [FFS88], [GMR89] und [MOV97, Kap. 10.4].

In der Praxis der Informationstechnologie (IT) werden Benutzer sehr häufig durch statische Passwortsysteme identifiziert (authentifiziert). Neben der Tatsache, dass das statische Passwort des Teilnehmers **A** häufig leicht erraten oder durch ein trojanisches Pferd (böswilliges Computerprogramm) ausspioniert werden kann, leiden solche Systeme darunter, dass der Verifizierer in den Besitz des Passwortes kommt, und sich somit anschließend selbst als Teilnehmer **A** ausgeben kann *(impersonation)*. Besser sind hier interaktive Wissens-Beweissysteme *(interactive proof of knowledge,* siehe [FFS88], [GMR89]), mit der **A** zwar die Kenntnis seines Wissens (Geheimnis) nachweisen kann, ohne das Geheimnis oder eine durch den Verifizierer verwertbare Information zu verraten.

Ein interaktives Wissensbeweisprotokoll hat die Eigenschaften *Abgeschlossenheit (completeness)* und *Widerstandsfähigkeit (soundness)*.

Ein interaktives Beweissystem (interaktives Beweisprotokoll) ist *abgeschlossen (complete)*, wenn ein legitimer Verifizierer die Identität eines legitimen Beweisers mit sehr großer Wahrscheinlichkeit akzeptiert, d.h. die Wahrscheinlichkeit für eine Ablehnung ist vernachlässigbar ([MOV97, Kap. 10.4]). Ein interaktives Beweissystem ist *widerstandsfähig (sound)*, wenn ein probabilistischer Polynomzeitalgorithmus M mit folgender Eigenschaft existiert: Wenn ein Betrüger **E** sich bei einem legitimen Verifizierer **B** mit nicht-vernachlässigbarer Wahrscheinlichkeit als Teilnehmer **A** ausweisen kann, dann kann M benutzt werden, um von **E** ein zum Wissen von **A** äquivalentes Wissen zu lernen, welches mit sehr großer Wahrscheinlichkeit (d.h. das Gegenereignis tritt mit vernachlässigbar kleiner Wahrscheinlichkeit auf) dazu befähigt, sich anschließend bei legitimen Verifizierern als Teilnehmer **A** auszugeben (siehe [MOV97, Kap. 10.4]). Completeness sichert also die fast hundertprozentige Erkennungsrate eines legitimen Teilnehmers, während Soundness einen Betrug (Impersonation, siehe unten) verhindert, sofern der Angreifer keine Kenntnis des Geheimnisses des Teilnehmers **A** bzw. einer äquivalenten Information besitzt. Wenn ein Betrüger **E** sich erfolgreich bei einem legitimen Verifizierer als Teilnehmer **A** ausgeben kann, obwohl das zugrundeliegende interaktive Beweissystem (Identifikationsverfahren) sound ist, so kann **E** mittels eines probabilistischen Polynomzeitalgorithmus auch **A**s Geheimnis oder eine äquivalente Information bestimmen (siehe [MOV97, Kap. 10.4]).

Wir fassen im Folgenden Identifikationsverfahren als interaktive Wissensbeweisprotokolle auf. Ein interaktives Wissensbeweisprotokoll hat die *Zero-Knowledge-Eigenschaft*, wenn das Protokoll durch einen Außenstehenden (ohne Kenntnis eines Geheimnisses) in folgendem Sinne simuliert werden kann: Es existiert ein probabilistischer Polynomzeitalgorithmus (Simulator), der ohne Interaktion mit dem richtigen Beweiser Sammlungen von Protokollnachrichten produziert, welche bzgl. ihrer Wahrscheinlichkeitsverteilung in Polynomzeit nicht unterschieden werden können von der Wahrscheinlichkeitsverteilung der Sammlungen der Nachrichten, die bei Durchführung des Protokolls durch einen legitimen Beweiser und einen beliebigen Verifizierer entstehen (siehe [FFS88], [MOV97, Kap. 10.4]). In einem Zero-Knowledge-Protokoll legt ein legitimer Beweiser also keine Information über sein Geheimnis offen, die nicht auch aus öffentlichen Informationen berechnet werden könnte. Dies gilt auch, wenn der legitime Beweiser mit einem betrügerischen Verifizierer kommuniziert, der dem Identifikationsprotokoll nicht folgt.

Angriffsziele

Impersonation. Ein Angreifer **E** weist sich erfolgreich gegenüber einem legitimen Verifizierer als Teilnehmer **A** aus.

Angriffsmethoden

Passive und aktive Angriffe (siehe Abschnitt 3.2.4). Ein Überblick über mögliche aktive Angriffe auf Identifikationsverfahren findet sich in [MOV97, Kap. 10.5]

Sicherheit von Identifikationsverfahren

Wir sagen, ein Identifikationsverfahren lege *keine verwertbare Information (no transferable information)* offen, wenn ein legitimer Beweiser **A** seine Identität bei einem legitimen Verifizierer

mit sehr großer Wahrscheinlichkeit nachweisen kann, aber keine Zusammenarbeit zwischen betrügerischem Beweiser und betrügerischem Verifizierer existiert, so dass sich ein betrügerischer Beweiser nach Durchführung polynomiell vieler Identifikationen zwischen legitimem Beweiser und betrügerischem Verifizierer mit nicht vernachlässigbarer Erfolgswahrscheinlichkeit als Teilnehmer **A** ausweisen kann. Eine verwertbare Information befähigt also einen betrügerischen Verifizierer dazu, sich anschließend als ein legitimer Beweiser **A** auszuweisen ([FFS88]).

Wir bezeichnen ein Identifikationsverfahren als *sicher*, wenn es keine verwertbaren Informationen offen legt ([FFS88]).

Ein Identifikationsverfahren mit der Zero-Knowledge-Eigenschaft legt keine verwertbaren Informationen offen, sofern das zugrunde liegende mathematische Problem schwierig zu lösen ist.

Ein stärkerer Sicherheitsbegriff für Identifikationsverfahren ist Sicherheit gegen Betrug unter der *gleichzeitigen Attacke (concurrent attack)*. Hierbei gesteht man dem Angreifer über die üblichen aktiven Angriffe (wie Einnahme der Rolle eines betrügerischen Verifizierers) hinaus zu, dass er als Verifizierer gleichzeitig mit vielen verschiedenen Kopien *(clones)* des Beweisers interagiert (siehe [BP02]). Bellare und Palacio [BP02] beweisen, dass das Schnorr-Identifikationsverfahren sicher gegen die gleichzeitige Attacke ist, vorausgesetzt, das in [BNPS01] eingeführte erweiterte diskrete Logarithmusproblem ist nicht effizient lösbar. Die Schwierigkeit dieses erweiterten diskreten Logarithmusproblems ist noch nicht hinreichend untersucht worden, daher ist hier Vorsicht geboten, wie auch die Autoren der zitierten Arbeit einräumen. Wir beschränken uns daher auf den oben eingeführten Sicherheitsbegriff für Identifikationsverfahren, dass das Protokoll keine verwertbare Information offen legt, obgleich sich das Ergebnis aus [BP02] vermutlich auf das von uns in Abschnitt 3.3.3.5 eingeführte Schnorr-Identifikationsverfahren übertragen lässt.

3.2.6 Pseudozufallszahlengeneratoren

Wir erläutern in diesem Abschnitt, wann wir einen Pseudozufallszahlengenerator sicher nennen. Formal ist die Sicherheit von Pseudozufallszahlengeneratoren in [Yao82] und [BM84] dargestellt. Eine gute Zusammenfassung bietet [Sti95, Kap. 12]. Wir erklären den Begriff informal.

In vielen kryptographischen Anwendungen müssen zur Laufzeit zufällige Zahlen oder Bit-Strings generiert werden. Guter (Pseudo-)Zufall kann durch physikalische Prozesse abgeleitet werden. Derartige Methoden sind allerdings zeit- und kostenintensiv. Ein *Pseudozufallszahlengenerator (Pseudo-random bit generator)* reduziert die Anzahl der benötigten zufälligen Bits in krytographischen Anwendungen. Hierzu startet ein Pseudozufallszahlengenerator mit einem zufälligen Bit-String, dem sogenannten *Seed*, und expandiert diesen in einen viel längeren Bit-String, der sozusagen zufällig aussieht.

Angriffsziele

Voraussage einer Bitfolge. Ein Angreifer **E** sagt eine vom Pseudozufallszahlengenerator ausgegebene Bitfolge erfolgreich voraus.

Angriffsmethoden

Passiver Angriff. Der Angreifer beobachtet eine Folge bislang vom Pseudozufallszahlengenerator ausgegebener Bits, um das nächste oder die nächsten auszugebenden Bits vorauszusagen.

Sicherheit von Pseudozufallszahlengeneratoren
Einen Pseudozufallszahlengenerator betrachten wir als *sicher*, wenn ein Angreifer (modelliert durch eine probabilistische Polynomzeit-Turing-Maschine) die Ausgabeverteilung des Pseudozufallszahlengenerators bei zufälligen Eingaben (Seeds) nicht von der Ausgabeverteilung einer echten Zufallsfolge (mit Gleichverteilung) unterscheiden kann.

Ein Pseudozufallszahlengenerator ist genau dann sicher, wenn ein Angreifer bei Kenntnis der Ausgabe der ersten $k-1$ Bits des Pseudozufallszahlengenerators nicht mit Wahrscheinlichkeit $> \frac{1}{2}$ das k-te Bit voraussagen kann (zum Beweis siehe z.B. [Sti95, Kap. 12]).

3.2.7 Bit Commitment-Verfahren

In der Realität kommt es vor, dass eine Person Alice ein wichtiges Dokument auf einem Briefbogen verfasst, den Briefbogen dann in einen Tresor legt, zu dem nur sie den Geheimcode zum Öffnen kennt. Dann übergibt Alice den Tresor an Bob. Obwohl Bob den Inhalt des Dokumentes nicht kennt, bis Alice den Tresor öffnet, würde man sagen, Alice habe sich vorab auf den Inhalt des Dokumentes *festgelegt (commited)*, denn sie kann ihn nachträglich nicht mehr ändern.

Übertragen auf die digitale Welt wollen wir annehmen, der Dokumenteninhalt sei ein Bit $b \in \{0,1\}$. Der Tresor entspricht einem Verschlüsselungsverfahren, das man auch *Bit Commitment-Verfahren* nennt. In der Regel handelt es sich hier um eine Funktion $BC : \{0,1\} \times X \to Y$, wobei X und Y endliche Mengen sind. Die Verschlüsselung des Bits b ist ein Wert $BC(b,x)$ mit $x \in X$ und wird auch *Blob* genannt. Will sich eine Person **A** auf einen Bitstring festlegen (commiten), so legt sie sich nacheinander auf die einzelnen Bits fest. Bit Commitment-Verfahren sollten folgende Eigenschaften besitzen:

a. Geheimhaltung (concealing) des Blobs Für ein Bit $b \in \{0,1\}$ kann ein Teilnehmer **B** das Bit b nicht probabilistisch in Polynomzeit aus dem Blob $BC(b,x)$ berechnen.

b. Verbindlichkeit (binding) für A **A** kann später den Blob „öffnen“, indem er den Wert von x offen legt. So kann **A** den Verifizierer **B** davon überzeugen, sich vorab tatsächlich auf das Bit b festgelegt zu haben. **A** ist probabilistisch in Polynomzeit nicht in der Lage, für einen Blob sowohl den Nachweis für $b = 0$ als auch den Nachweis für $b = 1$ führen zu können.

Angriffsziele

1. *Öffnen des Blobs* (Angriff auf Geheimhaltung). Der Angreifer **E** berechnet das Bit b, auf das sich ein Teilnehmer **A** festgelegt hat.

2. *Angriff auf Verbindlichkeit*. Der Angreifer **E** legt sich als Teilnehmer **A** vermeintlich auf ein Bit $b \in \{0,1\}$ fest, kann aber für seinen Blob später wahlweise sowohl den Nachweis für b als auch für $1-b$ führen.

Angriffsmethoden
Passive und aktive Angriffe (siehe Abschnitt 3.2.4).

Sicherheit von Bit Commitment-Verfahren
Wir nennen ein Bit Commitment-Verfahren *sicher*, wenn es die beiden Eigenschaften a. und b. besitzt.

3.3 Kryptoverfahren für algebraische Zahlkörper

In diesem Abschnitt stellen wir kryptographische Verfahren vor, die sich in Gruppen implementieren lassen, welche den Anforderungen aus Abschnitt 4.2 genügen, also beispielsweise in den Klassengruppen algebraischer Zahlkörper aus Abschnitt 4.3.

3.3.1 Designproblem für Kryptoverfahren über algebraischen Zahlkörpern

Wir wollen kryptographische Protokolle in Klassengruppen algebraischer Zahlkörper implementieren. Die meisten heute bekannten kryptographischen Protokolle, insbesondere Signaturverfahren wie RSA, ElGamal oder Varianten wie beispielsweise DSA, setzen die Kenntnis der Gruppenordnung voraus. Diese ist aber in unserem Kontext nicht bekannt und kann in der Praxis auch nicht effizient berechnet werden. Folglich war bis jetzt vollkommen unklar, wie man überhaupt kryptographische Verfahren konstruieren kann, die sich in algebraischen Zahlkörpern implementieren lassen.

Am Beispiel des klassischen DSA-Signaturverfahrens in $G = (\mathbb{Z}/p\mathbb{Z})^*$ verdeutlichen wir das Problem, bekannte kryptographische Protokolle auf endliche Abelsche Gruppen G mit unbekannter Ordnung (und unbekanntem Vielfachen der Ordnung) zu übertragen. Im Folgenden sei $M \in \{0,1\}^*$ sei der zu signierende Text.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle zufällig eine 160-Bit-Primzahl q, und wähle eine 1024-Bit-Primzahl p, so dass $q \mid p-1$;
wähle $\gamma_0 \in \mathbb{Z}/p\mathbb{Z}$ und berechne

$$\gamma = \gamma_0^{(p-1)/q} \bmod p \ , \tag{3.1}$$

und falls $\gamma = 1$, dann wähle ein anderes γ_0 (G ist die von γ erzeugte Untergruppe von $(\mathbb{Z}/p\mathbb{Z})^*$ der Ordnung q);
wähle zufällig eine ganze Zahl a mit $0 < a < q$ und berechne $\alpha = \gamma^a$.

As öffentlicher Schlüssel ist (p, q, γ, α), der private Schlüssel ist a.

Signatur: **A** führt die folgenden Schritte aus:
Wähle zufällig eine ganze Zahl k mit $0 < k < q$ und berechne $\varrho = \gamma^k \bmod p$;
berechne $r = \varrho \bmod q$ und

$$s = k^{-1}(h(M) + ar) \bmod q \ . \tag{3.2}$$

Die Signatur $Sig_{\mathbf{A},M}$ ist das Paar (s, r).

Verifikation: **B** führt die folgenden Schritte aus:
Prüfe, ob $0 < s < q$;
berechne $w = s^{-1} \bmod q$, $u_1 = wh(M) \bmod q$ und $u_2 = wr \bmod q$;

berechne $v = (\gamma^{u_1}\alpha^{u_2} \bmod p) \bmod q$.

B akzeptiert $Sig^{\mathsf{DSA}}_{\mathbf{A},M}$, wenn $v = r$.

Ohne die Kenntnis von $|G|$ kann weder (3.1) noch (3.2) berechnet werden. Mit (3.1) wird sicher gestellt, dass γ ein Erzeuger von G ist. Falls G zyklisch ist, dann ist die Wahrscheinlichkeit sehr groß, dass ein zufälliges $\gamma \in G$ ein Erzeuger von G oder einer sehr großen Untergruppe davon ist. In diesem Fall ist die zufällige Auswahl von γ ausreichend. Bei der Übertragung eines solchen Kryptoprotokolls auf endliche Abelsche Gruppen mit unbekannter Ordnung (wie Klassengruppen algebraischer Zahlkörper) war vollkommen unklar, wie mit (3.2) verfahren werden soll, wo letztlich die Exponenten modulo der (Unter-)Gruppenordnung reduziert werden. Für dieses Problem haben wir verschiedene Lösungsstrategien ausgearbeitet:

1. Wir ersetzen q durch eine beliebige große Zahl q' (in der Praxis eine Primzahl) und reduzieren die Exponenten modulo dieser Zahl q'. Dabei muss q' nicht notwendigerweise ein Teiler von $|G|$ sein.

   Der geschilderte Trick funktioniert, weil $s = x \bmod q$ gleichbedeutend ist mit $s = x - \ell q$ für eine ganze Zahl ℓ. Da im Allgemeinen $q' \neq |\langle\gamma\rangle|$ sein wird, ist dann auch $s \not\equiv x \pmod{|\langle\gamma\rangle|}$. Daher benötigt man zur Verifikation einer Signatur den zusätzlichen Korrektur-Faktor γ^{ℓ} als Teil der Signatur.

   Diesen Ansatz werden wir zum Entwurf des RDSA-Signaturverfahrens in Abschnitt 3.3.4.1 verfolgen. Er ist Gegenstand unserer Arbeit [BBHM02]. Die genannte Grundidee verfolgen wir auch zur Konstruktion der folgenden neuen Kryptoverfahren über Gruppen unbekannter Ordnung: CU-RDSA (Abschnitt 3.3.5.2), Blind-RDSA (Abschnitt 3.3.5.3), R-Schnorr-Identifikation (Abschnitt 3.3.5.4).

   Bei diesem Ansatz muss man allerdings gegenüber den jeweiligen Originalverfahren Effizienzverluste in Kauf nehmen. Wir stellen diese Effizienzverluste am Beispiel von Signaturverfahren vor:

   - *Effizienzverlust beim Signieren.* Die Erzeugung einer Signatur erfordert nun zwei Exponentiationen (beim Originalverfahren eine).
   - *Längere Signatur.* Die Signatur enthält nun eine ganze Zahl und zwei Gruppenelemente (beim Originalverfahren bzw. bei dessen kanonischer Verallgemeinerung auf Gruppen mit bekannter Ordnung eine ganze Zahl und ein Gruppenelement, siehe [MOV97, Abschn. 11.5.2]).
   - *Effizienzverlust beim Verifizieren.* Die Überprüfung einer Signatur erfordert nun die Berechnung eines Potenzproduktes von drei Gruppenelementen (beim Originalverfahren Potenzprodukt von zwei Gruppenelementen).

2. Wir verzichten einfach auf die modulare Reduktion. Diese Idee wurde in [PS98] für das Schnorr-Signaturverfahren vorgeschlagen. Wir werden sie bei der Diskussion des Signaturverfahrens DSA-No-Subgroup in Abschnitt 3.3.4.2 verfolgen.

   Bei diesem Ansatz muss man ebenfalls gegenüber dem Originalverfahren Effizienzverluste in Kauf nehmen: Sowohl zum Erstellen als auch zum Überprüfen einer Signatur muss die Exponentiation mit viel größeren Exponenten durchgeführt werden.

3. Eine weitere Lösungsstrategie besteht darin, das Fiat-Shamir-Signaturverfahren [FS87] so zu modifizieren, dass es auf diskreten Logarithmen statt auf Quadratwurzeln basiert. Das resultierende DL-basierte Protokoll wird jedoch zu ineffizient: Waren im Original-Fiat-Shamir-Signaturverfahren t Quadrierungen von Gruppenelementen zum Signieren notwendig, so erfordert die Signaturerstellung in der DL-basierten Variante t Exponentiationen.
   Effizienter erscheint unsere Modifikation des Fiat-Shamir-Signaturverfahrens zum R-Feige-Fiat-Shamir-Protokoll (R-FFS) in Abschnitt 3.3.5.1. R-FFS basiert auf der Schwierigkeit, k-te Wurzeln zu berechnen, z.B. mit $k = 2$ oder $k = 3$. Die Signaturerstellung erfordert die Berechnung von durchschnittlich $t/2$ Gruppenoperationen (maximal t Gruppenoperationen) und einer k-ten Potenz.
   Analog konstruieren wir in Abschnitt 3.3.5.4 basierend auf dem Fiat-Shamir-Identifikationsverfahren [FS87] unser neues R-Fiat-Shamir-Identifikationsverfahren.

4. In der kryptographischen Literatur wurden doch wenige Kryptoverfahren vorgeschlagen, welche die Kenntnis der Gruppenordnung nicht benötigen. So übertragen wir beispielsweise mit dem Guillou-Quisquater-Signaturverfahren ein auf dem Wurzelproblem beruhendes Protokoll, welches nicht die Kenntnis der Gruppenordnung erfordert. Diese Idee verfolgen wir in Abschnitt 3.3.3.

3.3.2 Kategorisierung der Kryptoverfahren

In den folgenden Abschnitten stellen wir Kryptoverfahren vor, die sich in endlichen Abelschen Gruppen mit unbekannter Gruppenordnung implementieren lassen, sofern die in der Einleitung von Kapitel 3 genannten Basisoperationen effizient durchgeführt werden können. Dabei kategorisieren wir die von uns vorgestellten Verfahren wie folgt:

Kategorie A. Hierbei handelt es sich um bereits bekannte kryptographische Protokolle, die sich mit sehr geringfügigen oder sogar ohne Änderungen in endlichen Abelschen Gruppen mit unbekannter Gruppenordnung implementieren lassen (siehe Abschnitt 3.3.3).

Kategorie B. Unter dieser Kategorie stufen wir neue, in unserer Arbeitsgruppe entwickelte Kryptoverfahren ein, an deren Entwurf wir maßgeblich beteiligt waren. (siehe Abschnitt 3.3.4).

Kategorie C. Kryptoverfahren dieser Kategorie sind von uns vollkommen neu entwickelt worden (siehe Abschnitt 3.3.5).

3.3.3 Kryptoverfahren der Kategorie A

Wir beschreiben nun bereits bekannte kryptographische Protokolle, die sich mit sehr geringfügigen oder sogar ohne Änderungen in endlichen Abelschen Gruppen mit unbekannter Gruppenordnung implementieren lassen.

3.3.3.1 Signaturverfahren Guillou-Quisquater

In diesem Abschnitt stellen wir das Signaturverfahren Guillou-Quisquater (GQ) vor. Wir zeigen, dass das GQ-Signaturverfahren im Random Oracle Model (siehe Abschnitt 3.2) sicher gegen einen

Angriff mit angepasst ausgewählten Texten (adaptive-chosen-message attack) ist, sofern in der zugrunde liegenden Gruppe nicht effizient Wurzeln berechnet werden können. Schließlich treffen wir Aussagen über die Effizienz des Verfahrens.

Das Protokoll

Das Guillou-Quisquater-Signaturverfahren [GQ88] basiert auf dem mathematischen Problem, Wurzeln in primen Restklassengruppen $(\mathbb{Z}/N\mathbb{Z})^*$ zu berechnen, wobei $N = p \cdot q$ wie im RSA-Verfahren gewählt wird. Die Kenntnis der Gruppenordnung wird im Guillou-Quisquater-Signaturverfahren nicht benötigt. Daher lässt es sich auf endliche Abelsche Gruppen mit unbekannter Gruppenordnung übertragen.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle eine endliche Abelsche Gruppe G;
wähle ein zufälliges Element $\alpha \in G$;
wähle eine Zufallszahl n aus $\{1, \ldots, 2^t\}$ (für einen Sicherheitsparameter $t \in \mathbb{Z}_{>0}$, s.u.) und berechne $\nu = \alpha^{-n}$.

As öffentlicher Schlüssel ist (G, ν, n), der private Schlüssel ist α.

Signatur: **A** führt die folgenden Schritte aus:
Wähle ein zufälliges Element $\kappa \in G$;
berechne $\varrho = \kappa^n$;
berechne $s = h(M \| \varrho)$ und $\sigma = \kappa \alpha^s$.

Die Signatur des Textes M ist $Sig_{\mathbf{A},M} = (s, \sigma)$.

Verifikation: **B** berechnet $\phi = \sigma^n \nu^s$ und akzeptiert die Signatur $Sig_{\mathbf{A},M}$ genau dann, wenn $h(M \| \phi) = s$ ist.

Das GQ-Signaturverfahren ist abgeschlossen *(complete)*, denn gültige Signaturen werden durch Teilnehmer, die dem Protokoll folgen, immer akzeptiert.

Proposition 3.3.1
Wenn $\boldsymbol{A}$ *und* $\boldsymbol{B}$ *dem vorstehenden GQ-Protokoll folgen, dann ist die Verifikation immer erfolgreich.*

Beweis: Mit den Bezeichnungen von oben gilt

$$\phi = \sigma^n \nu^s = (\kappa \alpha^s)^n \alpha^{-sn} = \kappa^n = \varrho \ , \tag{3.3}$$

und daher ist $h(M \| \phi) = h(M \| \varrho) = s$. □

Sicherheit

Eine GQ-Signatur $Sig_{\mathbf{A},M} = (s, \sigma)$ heiße gültig, wenn (3.3) für $Sig_{\mathbf{A},M}$ erfüllt ist. Zur Unterscheidung von legitimen Signaturen durch **A** schreiben wir gefälschte Signaturen als $\tilde{Sig}_{\mathbf{A},M}$.

Bei näherem Betrachten der Schlüsselerzeugung des GQ-Verfahrens folgt sofort, dass die Sicherheit des Verfahrens auf dem Wurzelproblem basiert:

Proposition 3.3.2
Ein Angreifer $\mathbf{E}$*, der in Polynomzeit das Wurzelproblem lösen kann, kann gültige Signaturen für beliebige Texte in Polynomzeit ohne Kenntnis des privaten Schlüssels erzeugen.*

Bemerkung
Die Bedeutung der Einwegfunktion h. Von entscheidender Bedeutung für die Sicherheit des GQ-Signaturverfahrens ist, dass die Funktion h in der Praxis tatsächlich die Einwegeigenschaft hat. Andernfalls sind beispielsweise folgende existenzielle Fälschungen (existential forgeries) möglich:

1. Wähle $\tilde{s}$ und κ zufällig, setze $\tilde{\sigma} = \kappa^{\tilde{s}}$ und berechne $\tilde{\varrho} = (\kappa^n \nu)^{\tilde{s}}$. Nach Voraussetzung war h keine Einweg-Funktion, d. h. es kann ein M berechnet werden, so dass $h(M \| \tilde{\varrho}) = \tilde{s}$ ist. Dann ist $(\tilde{s}, \tilde{\sigma})$ eine gültige GQ-Signatur für dieses M.

2. Wähle $\tilde{s}$ und κ zufällig, wähle $y \in \{1, \ldots, B_y\}$ für ein noch zu bestimmendes B_y, setze $\tilde{\sigma} = \kappa^{\tilde{s}y}$ und berechne $\tilde{\varrho} = (\kappa^{ny} \nu)^{\tilde{s}}$. Nach Voraussetzung war h keine Einweg-Funktion, d. h. es kann ein M berechnet werden, so dass $h(M \| \tilde{\varrho}) = \tilde{s}$ ist. Dann ist $(\tilde{s}, \tilde{\sigma})$ eine gültige GQ-Signatur für dieses M.

In unserer Arbeit [BBHM99] zeigen wir:

Satz 3.3.3
Sei G *eine endliche Abelsche Gruppe. Unter der Annahme, dass es keinen Algorithmus gibt, der das Wurzelproblem in* G *mit nicht vernachlässigbarer Wahrscheinlichkeit in polynomiell beschränkter Zeit berechnet, ist GQ in* G *im Random Oracle Model sicher.*

Nach heutigem Kentnisstand ist es in der Praxis unmöglich, n-te Wurzeln in Gruppen zu berechnen, die den Bedingungen aus Abschnitt 4.2 genügen. Hieraus folgt die Sicherheit des GQ-Signaturverfahrens.

Effizienz

Die Berechnung einer Signatur im GQ-Verfahren erfordert zwei Exponentiationen und eine weitere Gruppenoperation. Bei einer der beiden Exponentiationen können zur Effizienzsteigerung geeignete Potenzen vorberechnet werden. Bei der anderen Exponentiation ist dies nicht möglich. Das GQ-Verfahren erfordert zudem die Auswahl eines zufälligen Elementes in der Gruppe.

Die Verifikation einer GQ-Signatur erfordert zwei Exponentiationen, eine weitere Gruppenoperation und einen Gleichheitsentscheid.

Empfehlung zur Parameterwahl

In Analogie zur Wahl der Untergruppenordnung im Original-DSA-Verfahren [18694] kann man den öffentlichen Exponenten $n \in \{1, \ldots, 2^{160}\}$ wählen sowie die anerkannte kryptographische Hashfunktion RIPEMED-160$= h : \{0,1\}^* \to \{1, \ldots, 2^{160}\}$, z.B. RIPEMD-160 (siehe [MOV97, S. 349ff.]).

3.3.3.2 Gruppensignaturen

Chaum und van Heyst [CvH91] führten das Konzept von *Gruppensignaturen* ein. Ein Gruppensignaturverfahren hat folgende Eigenschaften:

1. Nur die Mitglieder einer vorab definierten Gruppe können eine gültige Signatur für eine Nachricht ausstellen.
2. Jeder kann die Gültigkeit einer Gruppensignatur überprüfen.
3. Niemand kann feststellen, welches der Gruppenmitglieder eine Signatur ausgestellt hat.
4. Im Streitfall kann die Signatur „geöffnet“ werden, d.h. die Identität der signierenden Person kann (ggf. mit Hilfe der Gruppenmitglieder) festgestellt werden.

Die in [CvH91, Abschn. 2] vorgestellte Methode lässt sich problemlos auf alle ElGamal-ähnlichen Signaturverfahren anwenden. Insbesondere erhält man auf diese Weise Gruppensignaturverfahren für RDSA und DSA-No-Subgroup.

Die grundlegende Idee lautet wie folgt: Das Gruppenmitglied i erzeugt sein eigenes Geheimnis a_i und sendet γ^{a_i} zum sogenannten Gruppenmanager. Der Gruppenmanager speichert zu jedem Teilnehmer i dessen öffentliche Information γ^{a_i}. In regelmäßigen kurzen Abständen (z.B. wöchentlich) sendet der Gruppenmanager jedem Gruppenmitglied eine Zufallszahl r_i und publiziert die Liste der temporären öffentlichen Schlüssel $(\gamma^{a_i})^{r_i}$, ohne dabei eine Zuordnung zu den i Teilnehmern erkennen zu lassen. In der vorgesehenen kurzen Zeitspanne verwendet der Teilnehmer i den temporären privaten Schlüssel $a_i r_i$.

Die resultierenden Gruppensignaturverfahren haben folgende positive Eigenschaften: Auch der Gruppenmanager kann keine Signaturen fälschen. Jedes Gruppenmitglied muss dauerhaft nur einen tatsächlich geheimen Schlüssel speichern; dieser könnte beispielsweise auf einer Smart Card gespeichert sein. Dieses Geheimnis wird mittels des temporären Wertes r_i cachiert. Ein versehentliches Offenlegen von r_i gefährdet in keiner Weise die Geheimhaltung des dauerhaften Geheimnisses a_i.

Eine exakte Definition von Gruppensignaturverfahren sowie tiefer gehende Literaturhinweise finden sich in [CS97].

3.3.3.3 Verschlüsselungsverfahren – DHIES

In diesem Abschnitt beschreiben wir, wie man Public-Key-Verschlüsselungsverfahren in endlichen Abelschen Gruppen mit unbekannter Ordnung implementieren kann, die den Anforderungen aus Abschnitt 4.2 genügen, also z.B. in den Klassengruppen algebraischer Zahlkörper aus Abschnitt 4.3. Bei der Übertragung traditioneller Public-Key-Verschlüsselungsverfahren stößt man auf folgende Schwierigkeiten:

- Etliche Verfahren wie z.B. RSA erfordern die Kenntnis der Gruppenordnung zum Entschlüsseln (siehe auch Abschnitt 3.3.1). Diese ist im Falle unserer Klassengruppen unbekannt.
- Die meisten Verfahren erfordern die Einbettung (injektive Abbildung) der zu verschlüsselnden Texte in Gruppenelemente. Hierbei muss aus Sicherheitsgründen eine Gleichverteilung vorliegen. Es ist bis heute aber vollkommen unklar, wie man in unserem Falle Texte in Idealklassen geeignet einbettet.

Diffie-Hellman Integrated Encryption Scheme (DHIES)

Wir beschreiben nun das Public-Key-Verschlüsselungsverfahren Diffie-Hellman Integrated Encryption Scheme (DHIES) aus [ABR01]. DHIES erfordert weder die Kenntnis der Gruppenordnung noch die Einbettung von Texten in Gruppenelemente. Daher kann DHIES in Klassengruppen algebraischer Zahlkörper implementiert werden.

Das Protokoll

Allen Teilnehmern wird die Funktion $deriveKey: G \times G \to \{0,1\}^*$ bekannt gemacht.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle eine endliche Abelsche Gruppe G;
wähle ein zufälliges Element $\gamma \in G$;
wähle eine Zahl L in der Größenordnung der Gruppenordnung: $L \approx |G|, L \geqslant |G|$;
wähle eine Zufallszahl $a \in \{1, \ldots, L-1\}$;
berechne $\alpha = \gamma^a$.

A hat den öffentlichen Schlüssel (G, L, γ, α) und den geheimen Schlüssel a.

Verschlüsselung: **B** führt die folgenden Schritte aus:
Wähle eine Zufallszahl $k \in \{1, \ldots, L-1\}$;
berechne $\varrho = \gamma^k$ und $\sigma = \alpha^k$;
berechne $K = deriveKey(\sigma, \varrho)$;
berechne K_{ENC} und K_{MAC} aus K;
berechne $EM = ENC(M, K_{ENC})$;
berechne $T = MAC(EM, K_{MAC})$.

Die Verschlüsselung des Textes M ist $Cipher_{\mathbf{A},M} = (\varrho, EM, T)$.

Entschlüsselung: **A** führt die folgenden Schritte aus:
berechne $\sigma = \varrho^a$;
berechne $K = deriveKey(\sigma, \varrho)$;
berechne K_{ENC} und K_{MAC} aus K;
berechne $T' = MAC(EM, K_{MAC})$.

Falls $T' \neq T$ verwerfe $Cipher_{\mathbf{A},M}$, andernfalls berechne den Klartext $M = ENC^{-1}(EM, K_{ENC})$.

Der folgende Satz zeigt die Vollständigkeit *(completeness)* des DHIES-Verschlüsselungsverfahrens, d.h. jeder DHIES-verschlüsselte Klartext kann wieder eindeutig rekonstruiert werden.

Proposition 3.3.4
*Wenn **A** und **B** dem vorstehenden DHIES-Protokoll folgen, dann kann aus der Verschlüsselung eines jeden Klartexts derselbe Klartext immer eindeutig rekonstruiert werden.*

Beweis: Mit den oben gemachten Bezeichnungen gilt offenbar

$$\alpha^k = \varrho^a = \sigma \ . \tag{3.4}$$

Also sind die Schlüssel K_{MAC} und K_{ENC} bei der Verschlüsselung und der Entschlüsselung jeweils identisch. Demnach gilt $ENC^{-1}(EM, K_{ENC}) = ENC^{-1}(ENC(M, K_{ENC}), K_{ENC}) = M$ und $T' = MAC(EM, K_{MAC}) = T$. □

Sicherheit

Die Sicherheitsanalysen des DHIES-Verfahrens aus [ABR01] gelten auch in endlichen Abelschen Gruppen mit unbekannter Ordnung, also insbesondere in Klassengruppen algebraischer Zahlkörper. DHIES ist sicher gegen Angepasst-Gewählte-Schlüsseltext-Angriffe (adaptive-chosen-ciphertext attacks), wenn *deriveKey*, MAC und ENC geeignet gewählt werden:

Satz 3.3.5
Seien deriveKey, ENC und MAC gegeben mit folgenden Eigenschaften:

1. *Gegeben γ^a, γ^k und γ^{ak}. Die Wahrscheinlichkeit für den Erfolg einer Unterscheidung von $deriveKey(\gamma^{ak}, \gamma^k)$ von einem zufällig gewählten Bit-String ist vernachlässigbar.*
2. *Die Wahrscheinlichkeit für den Erfolg eines Angriffs auf ENC mittels ausgewählten Klartexten (Bekannter-Klartext-Angriff, Known-Plaintext-Attack) ist vernachlässigbar.*
3. *Die Wahrscheinlichkeit für den Erfolg eines Angriffs auf MAC mittels ausgewählten Texten ist vernachlässigbar.*

Dann ist die Wahrscheinlichkeit für den Erfolg einer Unterscheidung mittels eines Angepasst-Gewählten-Schlüsseltext-Angriffs vernachlässigbar.

Folglich ist DHIES in einer beliebigen endlichen Abelschen Gruppe G sicher gegen eine Unterscheidung mittels eines Angepasst-Gewählten-Schlüsseltext-Angriffs, wenn das Diffie-Hellman-Problem (DHP) in G nur mit vernachlässigbarer Wahrscheinlichkeit effizient lösbar ist.

Effizienz

Da die in der Praxis gängigen symmetrischen Kryptosysteme (im Vergleich zu Public-Key-Verfahren) extrem effizient sind, vernachlässigen wir diese bei den Effizienzbetrachtungen von DHIES. Zur Verschlüsselung einer Nachricht mittels DHIES muss man zwei Exponentiationen mit Exponenten in der Größenordnung der Gruppenordnung ausführen. Beide Exponentiationen könnten komplett vorberechnet werden, sofern die Anwendung die Vorabgenerierung des zufälligen Exponenten erlaubt. Ist dies nicht möglich, können zumindest jeweils die Techniken der schnellen Exponentiation (siehe Abschnitt 5) mit entsprechenden Vorberechnungen von Potenzen angewendet werden. Zur Entschlüsselung ist eine Exponentiation mit einem Exponenten in der Größenordnung der Gruppenordnung auszuführen. Hierbei ist keine Form der Vorberechnung möglich.

3.3.3.4 Schlüsselaustauschverfahren: Diffie-Hellman-Varianten und weitere Protokolle

In diesem Abschnitt diskutieren wir einige Schlüsselaustauschverfahren, die sich sicher und effizient in den Gruppen implementieren lassen, welche den Anforderungen aus Abschnitt 4.2 genügen.

Diffie-Hellman (DH)

Das Protokoll

Vorab wird für alle Teilnehmer eine gemeinsam genutzte endliche Abelsche Gruppe G festgelegt sowie eine gute Näherung $L \approx |G|$ mit $|G| \leqslant L$, $L \in \mathbb{Z}_{>0}$;

Schlüsselerzeugung: Eine vertrauenswürdige Instanz (Trust Center) führt initial und einmalig folgende Schritte aus:
Sie wählt eine endliche Abelsche Gruppe G;
sie berechnet eine gute Näherung $L \approx |G|$ mit $|G| \leqslant L$, $L \in \mathbb{Z}_{>0}$;
sie wählt ein zufälliges Element $\gamma \in G$;
sie veröffentlicht G, L und γ für alle Teilnehmer.

Schlüsselvereinbarung: **A** und **B** führen die folgenden Schritte aus:
A wählt eine Zufallszahl a aus $\{1, \ldots, L\}$ und sendet γ^a an **B**;
B wählt eine Zufallszahl b aus $\{1, \ldots, L\}$ und sendet γ^b an **A**;
B erhält γ^a und berechnet den gemeinsamen Schlüssel mittels $K = (\gamma^a)^b$;
A erhält γ^b und berechnet den gemeinsamen Schlüssel mittels $K = (\gamma^b)^a$.

Sicherheit

Die Sicherheit des Diffie-Hellman-Schlüsselvereinbarungsprotokolls basiert auf dem Diffie-Hellman-Problem (DHP, siehe Abschnitt 3.1). Das Diffie-Hellman-Protokoll ist sicher gegen passive Angriffe, sofern das Diffie-Hellman-Problem nicht effizient lösbar ist. Andernfalls kann ein mit dem Diffie-Hellman-Protokoll vereinbarter geheimer Schlüssel unmittelbar durch einen passiven Angreifer berechnet werden.

Nach heutigem Kenntnisstand ist es in der Praxis unmöglich, das Diffie-Hellman-Problem in Gruppen zu lösen, die den Bedingungen aus Abschnitt 4.2 genügen.

Das Diffie-Hellman-Protokoll ist (in der beschriebenen Variante) nicht sicher gegen aktive Angreifer; denn folgende *Man-in-the-middle attack* bricht das Diffie-Hellman-Verfahren, d.h. berechnet einen zwischen **A** und **B** vereinbarten geheimen Schlüssel: Der Angreifer **E** schaltet sich aktiv in den Kommunikationskanal ein, indem er die zwischen **A** und **B** ausgetauschten Nachrichten abfängt und durch selbst erstellte Nachrichten ersetzt. Hierzu wählt er zwei zufällige Zahlen $\tilde{a}$, $\tilde{b}$ und berechnet $\gamma^{\tilde{a}}$, $\gamma^{\tilde{b}}$. Die von **A** an **B** gesendete Nachricht γ^a fängt **E** ab und schickt statt dessen $\gamma^{\tilde{a}}$ an **B** (welcher glaubt, die Nachricht sei von **A**). Ebenso ersetzt **E** die von **B** an **A** gesendete Nachricht γ^b durch $\gamma^{\tilde{b}}$. Anschließend glaubt **A**, $\gamma^{a\tilde{b}}$ sei der gemeinsame geheime Schlüssel; **B** hält $\gamma^{\tilde{a}b}$ für den gemeinsamen geheimen Schlüssel. **E** kann beide Schlüssel berechnen.

Weiter unten skizzieren wir, wie man durch Modifikation des Diffie-Hellman-Protokolls auch derartige aktive Angriffe abwehren kann.

Effizienz

Wir diskutieren die Effizienz des Diffie-Hellman-Protokolls.

Berechnungskomplexität Beide Kommunikationsteilnehmer haben jeweils zwei Exponentiationen durchzuführen. Die erste (zur Basis γ) kann komplett vorberechnet werden. (Wenn die Wahl der Zufallszahl erst online erfolgen soll, kann die erste Exponentiation mit den Techniken von Vorberechnung und schneller Exponentiation erfolgen, siehe Abschnitt 5.3.) Die zweite Exponentiation kann jeweils mittels schneller Exponentiation erfolgen; Vorberechnungen sind hier nicht möglich.

Kommunikationslast Es werden zwei Nachrichten übertragen, die jeweils aus einem Gruppenelement bestehen.

Diffie-Hellman-Varianten

In den Gruppen, die den Anforderungen aus Abschnitt 4.2 genügen (also insbesondere in den von in Abschnitt 4.3 vorgestellten Klassengruppen), lassen sich zudem die folgenden Schlüsselvereinbarungsprotokolle effizient und zumindest gegen passive Angriffe sicher implementieren. Alle vorgestellten Varianten basieren auf dem Diffie-Hellman-Protokoll.

ElGamal-Schlüsselaustausch. Beim ElGamal-Schlüsselaustauschverfahren (siehe [MOV97, S. 517]) wählt jeder Teilnehmer **B** in der Phase der Schlüsselerzeugung eine zufällige Zahl b, veröffentlicht $\beta = \gamma^b$ als öffentlichen Schlüssel und hält b geheim. Will nun **A** mit **B** einen geheimen Schlüssel vereinbaren, besorgt **A** sich **B**s öffentlichen Schlüssel, wählt eine Zufallszahl x und sendet γ^x an **B**. Sowohl **A** als auch **B** können dann den gemeinsamen geheimen Schlüssel γ^{bx} berechnen.

Sicherheit. Das ElGamal-Schlüsselaustauschverfahren ist (wie das Diffie-Hellman-Protokoll) sicher gegen passive Angriffe, sofern das Diffie-Hellman-Problem nicht effizient lösbar ist. Es handelt sich dabei um ein einseitiges authentisches Schlüsselvereinbarungsprotokoll, sofern die Authentizität (Echtheit) des öffentlichen Schlüssels des Teilnehmers **B** sichergestellt ist. Letzteres kann beispielsweise durch Ausgabe von Zertifikaten durch das Trust Center gewährleistet werden. Die oben geschilderte aktive Man-in-the-middle attack wird hier bereits verhindert.

Berechnungskomplexität. Beide Kommunikationsteilnehmer haben jeweils eine Exponentiation durchzuführen. Teilnehmer **B** kann hierbei die Techniken der schnellen Exponentiation durch Vorberechnung entsprechender Potenzen nutzen (siehe Abschnitt 5.3). Teilnehmer **A** kann alle Berechnungen im voraus ausführen, wenn er in der Lage ist, seine Zufallszahlen vorab zu produzieren und bereits Kommunikationsteilnehmern zuzuordnen. Andernfalls kann **A** die Techniken der schnellen Exponentiation nutzen.

Kommunikationslast. Es wird eine Nachricht übertragen, die aus einem Gruppenelement besteht. Die Kommunikationslast ist also gegenüber dem Diffie-Hellman-Verfahren um eine Nachricht (ein Gruppenelement) reduziert.

Diffie-Hellman Key Predistribution. Während beim ElGamal-Schlüsselaustauschverfahren einer der beiden Kommunikationsteilnehmer *eine* fixierte Geheimzahl und *einen* fixierten öffentlichen Schlüssel verwendet, benutzen beim Diffie-Hellman Key Predistribution *beide* Kommunikationsteilnehmer ihre fixierte Geheimzahl und ihren fixierten öffentlichen Schlüssel (siehe z.B. [Sti95, S. 264ff.]). Bei Verwendung von Public-Key-Zertifikaten, die durch ein

Trust Center ausgegeben werden, entsteht ein beidseitiges authentisches Schlüsselvereinbarungsverfahren. Das Verfahren ist sicher gegen passive und aktive Angriffe, sofern das Diffie-Hellman-Problem nicht effizient lösbar ist.

Berechnungskomplexität und Kommunikationslast werden hier minimiert: Zur Laufzeit ist von beiden Kommunikationspartnern lediglich eine Exponentiation durchzuführen (Vorberechnung und schnelle Exponentiationstechnik möglich); es muss keine Nachricht übertragen werden. Wie das ElGamal-Schlüsselvereinbarungsprotokoll leidet auch dieses Verfahren darunter, dass es voraussetzt, dass vorab bereits authentische Informationen ausgetauscht werden müssen: Entweder muss der Teilnehmer seinen authentischen öffentlichen Schlüssel dem Trust Center mitteilen, oder das Trust Center müsste dem Benutzer auf sicherem Wege dessen geheime Zufallszahl mitteilen.

STS-Protokoll (Station-to-Station Protocol). Das Station-to-Station Protocol ([DvOW92]) ist ein beidseitiges authentisches Schlüsselvereinbarungsverfahren. Insbesondere wird hier die aktive Man-in-the-middle attack verhindert. Im STS-Protokoll wird ein beliebiges Signaturverfahren eingesetzt, um die Authentizität der übertragenen Nachrichten zu gewährleisten. Hier kann ein beliebiges der in dieser Arbeit vorgestellten Signaturverfahren verwendet werden. Optional findet eine symmetrische Verschlüsselung statt, um beidseitig zu überprüfen, dass der authentische Kommunikationspartner am Ende des Protokolls über den gleichen geheimen Schlüssel verfügt. Als symmetrisches Verschlüsselungsverfahren kann irgendein sicheres und effizientes Verfahren eingesetzt werden; wir empfehlen die Wahl eins der etablierten Verfahren Triple-DES, IDEA oder AES. Im STS-Protokoll müssen drei Nachrichten (bestehend aus mehreren Gruppenelementen und Bytesequenzen) ausgetauscht werden.

MTI-Protokolle (Matsumoto-Takashima-Imai). Die MTI-Protokolle ([MTI86]) sind beidseitige authentische Schlüsselvereinbarungsverfahren, die insbesondere resistent gegen aktive Man-in-the-middle attacks sind. In den MTI-Protokollen müssen die Teilnehmer selbst keine Signaturen erzeugen. Es werden lediglich zwei Nachrichten ausgetauscht (bestehend aus mehreren Gruppenelementen und Bytesequenzen).

Weitere Schlüsselaustauschprotokolle

ITU-T X.509-Authentifikation. Bereits mit den kryptographischen Primitiven Public-Key-Verschlüsselung und ggf. zudem digitalen Signaturverfahren lassen sich eine Reihe von Schlüsselaustauschprotokollen implementieren, wie in [MOV97, Abschn. 12.5] dargelegt wird. Beispielsweise können wir demnach mit unserem Public-Key-Verschlüsselungsverfahren DHIES und unserem digitalen Signaturverfahren RDSA die in ITU-T X.509 ([95995]) spezifizierten 2-Wege- und 3-Wege-Authentifikationen umsetzen, bei deren Ablauf die Kommunikationspartner neben der gegenseitigen Authentifikation einen geheimen Schlüssel vereinbaren.

SPX-Protokolle. Die SPX-Protokolle der Digital Equipment Corporation (DEC) (Originalreferenz [TAP90, TA91], DEC-Implementierung siehe [AT91], siehe auch [Sch94, S. 55 f.]) sind beidseitige authentische Schlüsselvereinbarungsverfahren; sie sind insbesondere resistent gegen aktive Man-in-the-middle attacks. Die SPX-Protokolle kombinieren Public-Key-Verfahren mit symmetrischen Verfahren. Sie setzen dabei die Existenz einer Trusted Third Party (Trust Center) voraus. Wir benutzen in den SPX-Protokollen folgende Algorithmen:

- als Signaturverfahren RDSA
- als Public-Key-Verschlüsselungsverfahren DHIES
- als symmetrisches Verfahren irgendein sicheres und effizientes Verfahren. Wir empfehlen die Wahl eins der etablierten Verfahren Triple-DES, IDEA oder AES.

Burmester-Desmedt Konferenz-Schlüsselaustausch. In einem *Konferenz-Schlüsselaustauschverfahren* vereinbaren $t \geqslant 2$ Teilnehmer simultan einen gemeinsamen geheimen Schlüssel. Eine mögliche Anwendung sind verschlüsselte Telefonkonferenzen, zu der initial ein gemeinsamer geheimer Schlüssel ausgehandelt wird.

Eine offensichtliche Methode für ein Konferenz-Schlüsselaustauschverfahren funktioniert wie folgt: Jeder Teilnehmer **A** vereinbart mit einem der oben vorgestellten Protokolle (z.B. Diffie-Hellman) einen geheimen Schlüssel K_A mit einem Trust Center. Das Trust Center generiert einen einzigen gemeinsamen Schlüssel K und teilt diesen – verschlüsselt mit dem jeweiligen Key des Teilnehmers (K_A für Teilnehmer **A**) – allen Teilnehmern mit. Diese Methode weist allerdings einige Nachteile auf. Sie erfordert beispielsweise die Existenz eines Online-Trust Centers. Zudem kann es auf Grund des hohen Kommunikations- und Berechnungsaufwandes beim Trust Center zu Engpässen kommen.

Eins der effizientesten Konferenz-Schlüsselaustauschverfahren ist das von Burmester und Desmedt [BD95] (siehe auch [MOV97, S. 528]). Dieses lässt sich in Gruppen, die den Anforderungen aus Abschnitt 4.2 genügen, (also z.B. in den in Abschnitt 4.3 vorgestellten Klassengruppen algebraischer Zahlkörper) sicher und effizient implementieren. Die Basisvariante des Verfahrens ist sicher gegen passive Angriffe, falls das Diffie-Hellman-Problem nicht effizient gelöst werden kann. Das Verfahren kann aber zu einem authentischen Konferenz-Schlüsselaustauschverfahren erweitert werden, welches auch aktiven Angriffen widersteht.

3.3.3.5 Identifikationsverfahren

Der sicheren Identifikation (Authentifikation) von Teilnehmern kommt in der praktischen IT-Sicherheit eine enorme Bedeutung zu. In den meisten Anwendungen weist sich ein Teilnehmer **A** heute noch durch Preisgabe seines statischen Geheimnisses aus (Eingabe des Passwortes). Hierdurch werden sowohl der Verifizierer als auch ein Angreifer, der die Kommunikation einmalig belauscht, in die Lage versetzt, sich anschließend erfolgreich als Teilnehmer **A** auszugeben. Mehr Sicherheit garantieren kryptographische Identifikationsverfahren.

In endlichen Abelschen Gruppen, die den Anforderungen aus Abschnitt 4.2 genügen, (also z.B. in den in Abschnitt 4.3 vorgestellten Klassengruppen algebraischer Zahlkörper) lassen sich folgende bekannte Identifikationsprotokolle ohne Protokollanpassungen implementieren:

1. Das in Abschnitt 3.3.5.2 vorgestellte Protokoll zum Zero-Knowledge-Beweis der Kenntnis eines diskreten Logarithmus. Dieses verwenden wir in Abschnitt 3.3.5.2 zur Konstruktion eines konvertierbaren nicht-abstreitbaren Signaturverfahrens.

2. Mit jedem NF-Public-Key-Verschlüsselungsverfahren (also insbesondere mit DHIES aus Abschnitt 3.3.3.3) kann man diverse Challenge-Response-Identifikationsprotokolle durchführen (siehe [MOV97, Abschn. 10.3.3]).

3. Das gleiche gilt für alle NF-Signaturverfahren. Mit diesen lassen sich beispielsweise die in ITU-T X.509 ([95995]) spezifizierten 2-Wege- und 3-Wege-Authentifikationen durchführen (siehe auch [MOV97, Abschn. 10.3.3]).

3.3.3.6 Faire Public-Key-Kryptosysteme

Auch in jüngster Vergangenheit sind die Diskussionen nicht abgebrochen, ob man mehr dem Wunsch der Bürger nach Privatsphäre und Vertraulichkeit digital übermittelter Daten nachkommen soll oder mehr dem Wunsch der Regierungen, im Zweifelsfall von Bürgern digital übermittelte verschlüsselte Nachrichten entschlüsseln zu können, um organisierte Kriminalität besser bekämpfen zu können. Der Wunsch der Bürger kann erfüllt werden, indem starke Verschlüsselungsverfahren mit entsprechenden Schlüssellängen benutzt werden, die in der Praxis nicht gebrochen werden können. Dem Wunsch der Regierung könnte z.B. durch gesetzliches Verbot starker Verschlüsselungsverfahren nachgekommen werden oder durch die gesetzliche Verpflichtung des Bürgers, seinen privaten Schlüssel an einer vertrauenswürdigen Stelle zu hinterlegen, auf die der Staat im Zweifelsfall Zugriff hat.

Micali [Mic93] schlug einen Ansatz vor, der beiden Standpunkten gerecht wird. Micali spricht daher von *fairen Public-Key-Kryptosystemen*. In fairen Public-Key-Kryptosystemen zerlegt ein Teilnehmer seinen privaten Schlüssel in Teilgeheimnisse. Die Teilgeheimnisse sendet er verschlüsselt an verschiedene Treuhänder (ein Teilgeheimnis pro Treuhänder). Wie in einem Secret Sharing Scheme (siehe Abschnitt 3.3.3.8) kann der private Schlüssel des Teilnehmers rekonstruiert werden, wenn alle Treuhänder (z.B. auf richterlichen Beschluss) das von ihnen verwaltete Teilgeheimnis mit einbringen. Die Korrektheit der einzelnen Teilgeheimnisse kann von den Treuhändern nachgeprüft werden, ohne den privaten Schlüssel des Teilnehmers zu rekonstruieren. Somit werden betrügerische Teilnehmer ertappt, die den Treuhändern statt eines Teilgeheimnisses einen Zufallsstring senden. Im Folgenden machen wir deutlich, dass sich faire Schlüsselaustausch- und Verschlüsselungsverfahren auch in endlichen Abelschen Gruppen implementieren lassen, die den Anforderungen aus Abschnitt 4.2 genügen, also z.B. in den Klassengruppen algebraischer Zahlkörper aus Abschnitt 4.3. Das Original-Verfahren [Mic93] arbeitet in der primen Restklassengruppe modulo einer Primzahl p. Wir formulieren das Protokoll für beliebige endliche Abelsche Gruppen G.

Fairer ElGamal-Schlüsselaustausch und faire DHIES-Verschlüsselung

Fairer ElGamal-Schlüsselaustausch und faire DHIES-Verschlüsselung funktionieren mit $n \geqslant 2$ Treuhändern wie folgt:

1. Beim ElGamal-Schlüsselaustausch und DHIES-Verschlüsselungsverfahren benutzen die Teilnehmer eine endliche Abelsche Gruppe G mit Approximation L an die Gruppenordnung, $L \approx |G|, L \geqslant |G|$, und ein Basiselement γ.
   Der private Schlüssel eines Teilnehmers **A** sei a, der öffentliche Schlüssel sei $(G, L, \gamma, \alpha = \gamma^a)$.

2. **A** wählt n (bis auf a_n) zufällige Zahlen $a_1, \ldots, a_n \in \{\pm 1, \ldots, \pm(L-1)\}$ mit $a = a_1 + \ldots + a_n$;
   A berechnet für $i \in \{1, \ldots, n\}$ die Elemente $\alpha_i = \gamma^{a_i}$.

3. **A** sendet dem Treuhänder i ($i \in \{1, \ldots, n\}$) (verschlüsselt mit dem Public Key des Treuhänders) das Teilgeheimnis a_i mit der zugehörigen öffentlichen Information α_i;
   A sendet dem Trust Center den öffentlichen Schlüssel α.

4. Der Treuhänder i verifiziert, dass $a_i \in \{\pm 1, \ldots, \pm(L-1)\}$ und $\alpha_i = \gamma^{a_i}$;
   verläuft die Verifikation erfolgreich, sendet der Treuhänder die öffentliche Information $(\mathbf{A}, \alpha_i)$ (signiert) zum Trust Center;
   der Treuhänder hinterlegt bei sich (signiert) die geheime Information $(\mathbf{A}, a_i)$.

5. Wenn das Trust Center alle n öffentlichen Teilinformationen zum Teilnehmer **A** erhalten hat, überprüft es, ob $\alpha = \alpha_1 \ldots \alpha_n$. Falls dies der Fall ist, bescheinigt das Trust Center dem Teilnehmer **A** die Gültigkeit dieses öffentlichen Schlüssels α (z.B. durch Erstellung eines Zertifikates).

6. Ansonsten folgen die Kommunikationsteilnehmer **A** und **B** den Vorschriften des ElGamal-Schlüsselaustauschprotokolls bzw. des DHIES-Verschlüsselungsverfahrens.

Für weitere Diskussionen über faire Public-Key-Kryptosysteme verweisen wir auf [Mic93].

3.3.3.7 Pseudozufallszahlengeneratoren

Wir stellen in diesem Abschnitt zwei Pseudozufallszahlengeneratoren vor, die sich effizient in den Gruppen implementieren lassen, welche den Anforderungen aus Abschnitt 4.2 genügen.

Verallgemeinerter RSA-Pseudozufallszahlengenerator

Der folgende Pseudozufallszahlengenerator ergibt sich durch Verallgemeinerung des RSA-Pseudozufallszahlengenerators, welcher in [ACGS88] diskutiert wird.

Setup: Wähle eine endliche, Abelsche Gruppe G;
bestimme eine Approximation $L \approx |G|$ mit $L \geqslant |G|$;
wähle einen festen Exponenten $e \in \{2, \ldots, L-1\}$;
wähle eine kryptographische Hashfunktion $h : G \to \{1, \ldots, 2^{160}\}$.

Seed: Wähle ein zufälliges Element $\gamma_0 \in G$.

Biterzeugung: for $i = 1$ to ℓ do
$\gamma_i = \gamma_{i-1}^e$;
$z_i =$ niederwertigstes Bit von $h(\gamma_i)$.

Ausgabe: $z_1, \ldots, z_\ell$

In der Praxis kann als Hashfunktion eine der in kryptographischen Anwendungen gängigen Funktionen wie RIPEMD-160 (siehe [MOV97, S. 349ff.]) verwendet werden, falls die Gruppenelemente als Binärstrings kodierbar sind. Zudem empfehlen wir aus Effizienzgründen die Wahl von $e \in \{2, 3\}$. Zwecks weiterer Effizienzsteigerung können in einem „Schritt der Biterzeugung“ nicht nur ein einzelnes Bit, sondern bis zu 160 Pseudozufallsbits auf einmal erzeugt werden, indem eine entsprechende Anzahl von Bits des Hashwertes genommen wird.

Über die Sicherheit des verallgemeinerten RSA-Zufallszahlengenerators ist nichts bekannt.

Verallgemeinerter Blum-Micali-Pseudozufallszahlengenerator
Der folgende Pseudozufallszahlengenerator ergibt sich durch Verallgemeinerung des Blum-Micali-Pseudozufallszahlengenerators, welcher in [BM84] analysiert wird.

Setup: Wähle eine endliche, Abelsche Gruppe G;
wähle ein zufälliges Element $\gamma \in G$;
setze $L = \operatorname{ord} \gamma$;
wähle eine injektive Hashfunktion $h : \langle\gamma\rangle \to \{1, \ldots, L\}$.

Seed: Wähle eine Zufallszahl $x_0 \in D = \{1, \ldots, L\}$.

Biterzeugung: for $i = 1$ to ℓ do
$x_i = h(\gamma^{x_{i-1}})$ $(=: f(x_{i-1}))$;
$$z_i = \left\{ \begin{array}{ll} 1 & \text{falls } 0 \leqslant \log_\gamma h^{-1}(x_i) = x_{i-1} \leqslant \frac{L}{2} \\ 0 & \text{sonst} \end{array} \right\} (=: B(x_i)).$$

Ausgabe: $z_1, \ldots, z_\ell$

Das Originalprotokoll [BM84] arbeitet in der primen Restklassengruppe modulo einer Primzahl p, und $x_i = f(x_{i-1}) = \gamma^{x_{i-1}} \mod p$ kann als positive ganze Zahl aufgefasst werden. Wir verallgemeinern das Verfahren zur Implementierbarkeit in beliebigen endlichen Abelschen Gruppen, indem wir eine injektive Hashfunktion $h : \langle\gamma\rangle \to \{1, \ldots, L\}$ einführen.

Satz 3.3.6
Der verallgemeinerte Blum-Micali-Pseudozufallszahlengenerator ist sicher im Random Oracle Model, sofern das diskrete Logarithmusproblem in G nicht probabilistisch in Polynomzeit gelöst werden kann.

Beweis: Aus den Ausführungen von Blum und Micali [BM84] folgt unmittelbar, dass der verallgemeinerte Blum-Micali-Pseudozufallszahlengenerator sicher ist, wenn er die beiden folgenden Eigenschaften besitzt:

1. D ist eine endliche Menge, und $f : D \to D$ ist eine effizient berechenbare Permutation.
2. $B : D \to \{0, 1\}$, und $B(x)$ kann nicht probabilistisch in Polynomzeit berechnet werden, wenn nur $x \in D$ bekannt ist; $B(x)$ kann hingegen effizient berechnet werden, wenn $y = f^{-1}(x)$ gegeben ist.

Zu 1. Betrachte die Funktion $\tilde{f} : D \to \langle\gamma\rangle,\ x \mapsto \gamma^x$. Wegen $|D| = |\langle\gamma\rangle| = L$ und $D = \{1, \ldots, L\}$ ist $\tilde{f}$ bijektiv. Die Funktion $h : \langle\gamma\rangle \to D$ ist nach Voraussetzung injektiv und wegen $|D| = | < \gamma > |$ surjektiv. Die Funktion $f = h \circ \tilde{f}$ ist als Hintereinanderausführung zweier bijektiver Funktionen bijektiv, d.h. eine Permutation auf D.

Zu 2. Offenbar bildet B die Menge D auf $\{0, 1\}$ ab, und für $x \in D$ kann $B(x)$ bei gegebenem $y = f^{-1}(x)$ trivialerweise effizient berechnet werden, da $B(x) = 1$, falls $0 \leqslant y \leqslant \frac{L}{2}$, und $B(x) = 0$ andernfalls.
Es genügt daher, noch folgendes zu zeigen: Angenommen, es sei nur $x \in D$ bekannt, und dennoch sei $B(x)$ probabilistisch in Polynomzeit berechenbar. Dann kann das diskrete Logarithmusproblem zur Basis γ in G gelöst werden (im Widerspruch zur Voraussetzung).

Sei also nur $x \in D$ gegeben, und $B(x)$ sei probabilistisch in Polynomzeit berechenbar. Das bedeutet, $x_{i-1} = \log_\gamma h^{-1}(x_i)$ kann probabilistisch in Polynomzeit bestimmt werden für eine beliebige, aber fixe injektive Funktion $h : \langle\gamma\rangle \to \{1, \ldots, L\}$. Also sind $h^{-1}(x_i) =: \gamma'$ und $\log_\gamma \gamma'$ probabilistisch in Polynomzeit berechenbar, im Widerspruch zur Voraussetzung.

□

Zur Implementierung des verallgemeinerten Blum-Micali-Pseudozufallszahlengenerators in Klassengruppen algebraischer Zahlkörper wählt man L in der Größenordnung von $|G|$ und eine Hashfunktion $h : G \to \{1, \ldots, L\}$.

3.3.3.8 Secret Sharing

In einem *Secret Sharing-Verfahren* wird ein geheimer Schlüssel so auf verschiedene Teilnehmer aufgeteilt, dass der Schlüssel nur rekonstruiert werden kann, wenn eine vorher bestimmte Anzahl dieser Personen ihre Teilgeheimnisse zusammen bringt.

RDSA kann so modifiziert werden, dass eine Gruppe von n Personen ($n \geqslant 2$) einen öffentlichen Schlüssel besitzt und sich den privaten Schlüssel teilt, so dass dieser nur verwendet werden kann, wenn alle n Personen ihre Teilgeheimnisse zusammen bringen. Ein entsprechendes Verfahren nennt man auch *(n,n)-Treshold-System*.

Wir verwenden eine ähnliche Konstruktion wie die aus Abschnitt 3.3.3.6: Der Private Key a wird in n Summanden a_i gesplittet: $a = a_1 + \ldots + a_n$, $a_i \in \{\pm 1, \pm 2, \ldots, \pm k\}$, wobei $\{1, 2, \ldots, k\}$ der zulässige Bereich für private Schlüssel ist. Der öffentliche Schlüssel bleibt unverändert: $\alpha = \gamma^a$.

Ansonsten wird das RDSA-Protokoll wie üblich durchgeführt. Wir setzen voraus, dass die Einheit, welche von den n Teilnehmern die Eingabe ihrer Teilgeheimnisse einfordert, vertrauenswürdig ist, denn diese Einheit erhält Kenntnis des privaten Schlüssels.

3.3.3.9 Subliminal Channel

In einem *Subliminal Channel-Protokoll* wird in einem kryptographischen Protokoll (z.B. Signaturverfahren) ein verdeckter Kanal geöffnet, über den Nachrichten für einen dedizierten Kommunikationsteilnehmer kodiert werden können, ohne dass ein Beobachter der Kommunikation überhaupt weiß, dass hier geheime Nachrichten ausgetauscht werden (vgl. mit dem Ansatz der *Steganographie*). Häufig kennen die beiden Kommunikationsteilnehmer den gleichen privaten Schlüssel.

Wir beschreiben beispielhaft , wie man in der Ohta-Okamoto-Variante [OO90, OO88] des Guillou-Quisquater-Identifikationsverfahrens (siehe Abschnitt 3.3.5.2) und im DSA-No-Subgroup-Signaturverfahren aus Abschnitt 3.3.4.2 einen Subliminal Channel einbaut. In beiden Beispielen kann niemand außer dem Kommunikationspartner die versteckte Nachricht extrahieren. Außenstehende können nicht einmal deren Existenz bemerken. Wir gehen jeweils davon aus, dass **B** ebenfalls den privaten Schlüssel von **A** kennt und sich beide Kommunikationsteilnehmer einen symmetrischen Schlüssel K hinreichender Länge teilen. Die beiden Kommunikationsteilnehmer verwenden ein One-Time-Pad (wir verwenden bitweises XOR) mit dem Schlüssel K, um aus der Klartext-Nachricht einen scheinbar zufällig gewählten Bitstring (eine Zufallszahl) zu machen.

Subliminal Channel für das Ohta-Okamoto-Identifikationsverfahren
Der Verifizierer **B** kodiert die zu cachierende Nachricht $\tilde{M}$ in der Challenge: $e = \tilde{M} \oplus K$. Der Beweiser **A** erhält die cachierte Nachricht durch sofortige Entschlüsselung mittels $\tilde{M} = e \oplus K$. Die maximal übertragbare Bitlänge in $\tilde{M}$ ist die Bitlänge von $\lfloor \frac{L}{2} \rfloor$, wobei L eine Approximation der Gruppenordnung $|G|$ ist. Der symmetrische Schlüssel K muss mindestens ebenso lang sein.

Subliminal Channel für das DSA-No-Subgroup-Signaturverfahren
Der Signierer **A** kodiert die zu cachierende Nachricht $\tilde{M} \in \{1, \ldots, B_k\}$ im Exponenten k, der scheinbar zufällig gewählt wird: $k = \tilde{M} \oplus K$. Der Verifizierer **B** erhält **A**s Signatur (s, ϱ) für einen Text M, verifiziert diese und berechnet $k = s + ah(M\|\varrho)$, um die versteckte Nachricht mittels $\tilde{M} = k \oplus K$ zu bestimmen. Die maximal übertragbare Bitlänge in $\tilde{M}$ ist die Bitlänge von B_k. Der symmetrische Schlüssel K muss mindestens ebenso lang sein.

3.3.4 Kryptoverfahren der Kategorie B

Wir beschreiben nun neue, in unserer Arbeitsgruppe entwickelte Kryptoverfahren, an deren Entwurf wir maßgeblich beteiligt waren.

3.3.4.1 Signaturverfahren RDSA

In diesem Abschnitt stellen wir das Signaturverfahren RDSA vor. Zudem beleuchten wir dessen Sicherheit und Effizienz. Bei unserer Darstellung lehnen wir uns an unsere Arbeit [BBHM02] an.

Wir werden sehen: RDSA ist im Random Oracle Model sicher gegen einen Angriff mit angepasst ausgewählten Texten (adaptive-chosen-message attack). In unserer Arbeit [BBHM02] weisen wir nach, dass die Berechnung von Wurzeln in der zugrunde liegenden Gruppe und die existenzielle Fälschung von RDSA-Signaturen mit adaptive-chosen-message attacks polynomiell äquivalent sind. Bei geeigneter Wahl einer Klassengruppe eines algebraischen Zahlkörpers (siehe Abschnitt 4.3) können Wurzeln nicht effizient berechnet werden (siehe Abschnitte 4.2, 4.4). Hieraus folgt die Sicherheit des RDSA-Signaturverfahrens in den entsprechenden algebraischen Zahlkörpern.

Das Protokoll

Zur Konstruktion des RDSA-Signaturverfahrens modifizieren wir zunächst das in Abschnitt 3.3.1 vorgestellte klassische DSA-Signaturverfahren, um zur Effizienzsteigerung die modulare Invertierung in (3.2) zu vermeiden. Dazu wählen wir die DSA-Variante EG II.3 aus [HMP94] aus. Dann wird aus (3.2)

$$s = -ah(M\|\varrho) + k \bmod q \ , \tag{3.5}$$

mit entsprechender Modifikation der Signatur ($Sig_{\mathbf{A},M} = (s, \varrho)$) und Modifikation der Verifikations-Prozedur (siehe auch Abschnitt 3.3.4.2).

RDSA entsteht dann durch Einführung einer beliebigen großen Zahl q (in der Praxis eine Primzahl). Wir reduzieren die Exponenten modulo dieser Zahl q. Dabei muss q nicht notwendigerweise ein Teiler von $|G|$ sein.

Daraus resultiert das folgende RDSA-Signaturverfahren für beliebige endliche Abelsche Gruppen G. Dabei sei $h : \{0,1\}^* \to \{0, \ldots, q-1\}$ eine öffentlich bekannte kryptographische Hashfunktion, $M \in \{0,1\}^*$ sei der zu signierende Text.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:

Wähle eine endliche, Abelsche Gruppe G und eine hinreichend große zufällige Primzahl q mit $2^{t-1} < q \leqslant 2^t$ für einen Sicherheitsparameter $t \in \mathbb{Z}_{>0}$ (Empfehlung zur Parameterwahl siehe unten);
wähle eine gute Näherung $L \approx |G|$ mit $|G| \leqslant L$, $L \in \mathbb{Z}_{>0}$;
wähle ein zufälliges Element $\gamma \in G$;
wähle eine Zufallszahl a aus $\{2, \ldots, q-1\}$ und berechne $\alpha = \gamma^a$.

As öffentlicher Schlüssel ist (G, γ, α, q), der private Schlüssel ist a.

Signatur: **A** führt die folgenden Schritte aus:

Wähle eine Zufallszahl k aus $\{0, \ldots, qL-1\}$;
berechne $\varrho = \gamma^k$ und $x = -ah(M\|\varrho) + k$;
berechne ganze Zahlen s und ℓ, so dass $x = \ell q + s$ und $0 \leqslant s < q$;
berechne $\lambda = \gamma^\ell$.

Die Signatur des Textes M ist $Sig_{\mathbf{A},M} = (s, \varrho, \lambda)$.

Verifikation: **B** akzeptiert die Signatur $Sig_{\mathbf{A},M}$ genau dann, wenn $0 \leqslant s < q$ und $\gamma^s \alpha^{h(M\|\varrho)} \lambda^q = \varrho$ ist.

Das RDSA-Signaturverfahren ist abgeschlossen *(complete)*, denn gültige Signaturen werden durch Teilnehmer, die dem Protokoll folgen, immer akzeptiert.

Proposition 3.3.7
*Wenn **A** und **B** dem vorstehenden RDSA-Protokoll folgen, dann ist die Verifikation immer erfolgreich.*

Beweis: Mit den Bezeichnungen von oben gilt

$$\gamma^s \alpha^{h(M\|\varrho)} \lambda^q = \gamma^{-ah(M\|\varrho)+k-\ell q} \cdot \gamma^{ah(M\|\varrho)} \cdot \gamma^{\ell q} = \gamma^k = \varrho \ . \tag{3.6}$$

□

Sicherheit

Eine RDSA-Signatur $Sig_{\mathbf{A},M} = (s, \varrho, \lambda)$ heiße gültig, wenn (3.6) für $Sig_{\mathbf{A},M}$ erfüllt ist. Zur Unterscheidung von legitimen Signaturen durch **A** schreiben wir gefälschte Signaturen als $\tilde{Sig}_{\mathbf{A},M}$.

Wir zeigen nun, dass die Sicherheit von RDSA auf dem Wurzelproblem (root problem) basiert. (RDSA steht für „Root-based Digital Signature Algorithm“.)

Proposition 3.3.8
*Ein Angreifer **E**, der in Polynomzeit das Wurzelproblem lösen kann, kann gültige Signaturen für beliebige Texte in Polynomzeit ohne Kenntnis des privaten Schlüssels erzeugen.*

Beweis: Seien $\gamma, \alpha \in G$ und ein Text M gegeben. **E** wähle $\tilde{\varrho} \in G$ und $\tilde{s} \in \{0, \ldots, q-1\}$ beliebig und berechne $h(M\|\tilde{\varrho})$ und $\tau = \tilde{\varrho} \alpha^{-h(M\|\tilde{\varrho})} \gamma^{-\tilde{s}}$. Dann berechne **E** das Gruppenelement $\tilde{\lambda}$, die q-te Wurzel von τ. Da q eine zufällig gewählte, große Primzahl ist, ist die Wahrscheinlichkeit

vernachlässigbar klein, dass q ein Teiler von $|G|$ ist; falls τ keine q-te Wurzel hat, treffe **E** eine andere Wahl für $\tilde{\varrho}$ und $\tilde{s}$. $\widetilde{Sig}_{\mathbf{A},M} = (\tilde{s}, \tilde{\varrho}, \tilde{\lambda})$ ist eine gültige Signatur von **A** für M. □

Zur Wahrung der Sicherheit ist die Verwendung der kryptographischen Hashfunktion h notwendig. Würde man $h(M\|\varrho)$ durch M ersetzen, wären existenzielle Fälschungen möglich (siehe unsere Arbeit [BBHM02]).

Satz 3.3.9
Sei G eine endliche Abelsche Gruppe. Unter der Annahme, dass es keinen Algorithmus gibt, der das Wurzelproblem in G mit nicht vernachlässigbarer Wahrscheinlichkeit in polynomiell beschränkter Zeit berechnet, ist RDSA in G im Random Oracle Model sicher.

Beweis: Der Beweis findet sich in unserer Arbeit [BBHM02]. □

Nach heutigem Kentnisstand ist es in der Praxis unmöglich, q-te Wurzeln in Gruppen zu berechnen, die den Bedingungen aus Abschnitt 4.2 genügen.

Effizienz

Die Berechnung einer Signatur im RDSA-Verfahren erfordert zwei Exponentiationen (mit Exponenten $\approx 2^{t+\lceil \log L \rceil}$ bzw. 2^t, mit der Approximation $L \approx |G|$ und dem Sicherheitsparameter $t \in \mathbb{Z}_{>0}$), wobei zur Effizienzsteigerung geeignete Potenzen vorberechnet werden können. Die Exponentiation mit dem Exponenten $\approx 2^{t+\lceil \log L \rceil}$ kann sogar komplett vorberechnet werden.

Die Verifikation einer RDSA-Signatur erfordert eine simultane t-Bit-Exponentiation (welche naiv durch drei t-Bit-Exponentiationen und zwei weitere Gruppenoperationen durchgeführt werden könnte). Zudem hat der Verifizierer über die Gleichheit zweier Gruppenelemente zu entscheiden.

Empfehlung zur Parameterwahl

Beweisbare Sicherheit Ist in einer Signaturanwendung ein beweisbares Sicherheitsniveau[6] wichtig, so kann man im RDSA-Verfahren in Analogie zur Wahl der Untergruppenordnung im Original-DSA-Verfahren [18694] folgende Parameter wählen: $t = 160$. d.h. $q \sim 2^{160}$ prim, $L \approx |G|$, $|G| \leqslant L$, $L \in \mathbb{Z}_{>0}$, $k \in \{0, \ldots, qL-1\}$.

Verwendung kürzerer Exponenten Für die Praxis genügt es nach heutigem Kenntnisstand, $K = L \approx |G|$ sehr viel größer als $q \approx 2^{160}$ zu verwenden, also den Exponenten k zufällig aus $\{0, \ldots, L-1\}$ zu wählen. Der theoretische Nachweis der Sicherheit von RDSA im Random Oracle Model gelingt dann allerdings nicht mehr. Das liegt daran, dass man Werte s, ϱ und $h(M\|\varrho)$ und damit implizit eine Signatur vorgeben kann, für die es genau ein geeignetes $y \in \{1, \ldots, |G|\}$, aber kein $k \in \{1, \ldots, |G|\}$, sondern nur ein eindeutig modulo $q|G|$ bestimmtes $k > |G|$ gibt. Da der Signierer aber nach Voraussetzung niemals $k > |G|$ wählt, ergibt sich für den Signierer die Wahrscheinlichkeit 0, diese Signatur (s, ϱ, λ) zu erzeugen, während sich für den Simulator eine Wahrscheinlichkeit > 0 ergibt. Somit unterscheiden sich die beiden Ausgabeverteilungen von Signierer und Simulator voneinander. Der Reduktionsbeweis scheitert daher. Wir haben jedoch keinen Anhaltspunkt dafür, dass ein Angreifer aus dieser Tatsache in der Praxis Nutzen ziehen könnte.

[6] (basierend auf gewissen Annahmen)

In unserer Arbeit [BBHM02] schlagen wir zur Effizienzverbesserung die Verwendung kürzerer Exponenten $k \in \{0, \ldots, q-1\}$ vor. Hamdy empfiehlt in [Ham02, Abschnitt 6.7] sogar noch kürzere Exponenten. (Siehe hierzu auch die Originalarbeit [OW96] über die Verwendung kurzer Exponenten im Diffie-Hellman-Schlüsselaustauschprotokoll.) Es stellte sich allerdings heraus, dass bei Verwendung von Exponenten $k \in \{0, \ldots, q-1\}$ die Kenntnis weniger Klartext-Signatur-Paare genügt, um mittels LLL-Reduktion den privaten Schlüssel eines Teilnehmers zu ermitteln ([FP03]). Daher können die Empfehlungen für die Wahl eines kurzen Exponenten k aus [BBHM02, Ham02] heute nicht mehr gelten.

3.3.4.2 Signaturverfahren DSA-No-Subgroup

In diesem Abschnitt stellen wir das Signaturverfahren DSA-No-Subgroup vor. Zudem beleuchten wir dessen Sicherheit und Effizienz.

Das Protokoll

DSA-No-Subgroup entsteht durch Übertragung der Variante EG II.3 ([HMP94]) des DSA-Signaturverfahrens (Digital Signature Algorithm, [18694]) auf endliche Abelsche Gruppen G mit unbekannter Ordnung. In DSA- oder ElGamal-ähnlichen Signaturverfahren reduziert man typischerweise die in den Berechnungen vorkommenden Exponenten modulo der Gruppenordnung. Darauf wird in DSA-No-Subgroup einfach verzichtet. Ein entsprechender Vorschlag wurde erstmals in [PS98] für das Schnorr-Signaturverfahren geäußert. Die Kenntnis der Gruppenordnung wird damit nicht mehr benötigt. Allerdings muss man Exponentiationen mit sehr großen Exponenten durchführen. Dies führt zu Effizienzverlusten gegenüber dem klassischen DSA-Verfahren.

Es folgt die schematische Darstellung der DSA-Variante EG II.3 aus [HMP94] ohne modulare Reduktion des Exponenten. Dabei sei $h : \{0,1\}^* \to \{0, \ldots, B_h\}$ eine öffentlich bekannte kryptographische Hashfunktion, B_a, B_k, B_h seien positive ganze Zahlen, für deren Wahl wir weiter unten eine Empfehlung geben.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle eine endliche Abelsche Gruppe G;
wähle ein zufälliges Element $\gamma \in G$;
wähle eine Zufallszahl a aus $\{1, \ldots, B_a\}$ und berechne $\alpha = \gamma^a$.

As öffentlicher Schlüssel ist (G, γ, α), der private Schlüssel ist a.

Signatur: **A** führt die folgenden Schritte aus:
Wähle eine Zufallszahl k aus $\{1, \ldots, B_k\}$;
berechne $\varrho = \gamma^k$ und $s = -ah(M \| \varrho) + k$.

Die Signatur des Textes M ist $Sig_{\mathbf{A},M} = (s, \varrho)$.

Verifikation: **B** akzeptiert die Signatur $Sig_{\mathbf{A},M}$ genau dann, wenn $\gamma^s \alpha^{h(M\|\varrho)} = \varrho$ ist.

Das DSA-No-Subgroup-Signaturverfahren ist abgeschlossen *(complete)*, denn gültige Signaturen werden durch Teilnehmer, die dem Protokoll folgen, immer akzeptiert.

Proposition 3.3.10
*Wenn **A** und **B** dem vorstehenden DSA-No-Subgroup-Protokoll folgen, dann ist die Verifikation immer erfolgreich.*

Beweis: Mit den Bezeichnungen von oben gilt

$$\gamma^s \alpha^{h(M\|\varrho)} = \gamma^{-ah(M\|\varrho)+k} \cdot \gamma^{ah(M\|\varrho)} = \gamma^k = \varrho \ . \tag{3.7}$$

□

Sicherheit

Wir zitieren das Ergebnis [Ham02, Theorem 6.4.2]:

Satz 3.3.11
Seien $a \in \{1, \ldots, B_a\}$, $k \in \{1, \ldots, B_k\}$ zufällig gewählt, sei $h \in \{1, \ldots, B_h\}$ und sei B_n die maximale Anzahl von Texten, die mit einem festen Schlüssel signiert werden. Unter der Voraussetzung, dass $B_h B_a B_n / B_k$ und $1/B_h$ vernachlässigbar sind, gilt: Wenn die existentielle Fälschung (existential forgery) einer DSA-Signatur mit Hilfe eines Angepasst-gewählten-Text-Angriffs (adaptive chosen-message attack) in Polynomzeit eine nicht vernachlässigbare Erfolgswahrscheinlichkeit hat, dann können diskrete Logarithmen in G mit nicht vernachlässigbarer Wahrscheinlichkeit in Polynomzeit berechnet werden.

Nach heutigem Kentnisstand ist es in der Praxis unmöglich, diskrete Logarithmen in Gruppen zu berechnen, die den Bedingungen aus Abschnitt 4.2 genügen.

Effizienz

Zur Anwendbarkeit obigen Satzes muss $B_h B_a B_n / B_k$ vernachlässigbar klein sein. Demnach muss B_k sehr viel größer sein als beispielsweise im RDSA-Signaturverfahren (siehe Abschnitt 3.3.4.1). Wenn 2^{-t} als vernachlässigbar gilt, dann wird man $B_h = B_a = 2^{2t}$ wählen, und dann muss $B_k \geqslant 2^{5t}$ sein. Falls z.B. 2^{-80} als vernachlässigbar gilt und $B_h = B_a = 2^{160}$ ist, dann muss $B_k \geqslant 2^{400}$ sein.

Dies wirkt sich unmittelbar auf die Effizienz der Signatur-Erzeugung aus, da der zufällig gewählte Exponent k etwa 400 Bits lang sein muss, anstelle von etwa 160 Bits beim Original-DSA-Verfahren.

Die Berechnung einer Signatur im DSA-No-Subgroup-Verfahren erfordert eine Exponentiation, wobei zur Effizienzsteigerung geeignete Potenzen vorberechnet werden können.

Die Verifikation einer DSA-No-Subgroup-Signatur erfordert zwei Exponentiationen, eine weitere Gruppenoperation und einen Gleichheitsentscheid.

Empfehlung zur Parameterwahl

In Analogie zur Wahl der Untergruppenordnung im Original-DSA-Verfahren [18694] kann man den geheimen Exponenten a zufällig in $\{1, \ldots, 2^{160}\}$ wählen. Zudem kann man die anerkannte kryptographische Hashfunktion RIPEMD-160$= h : \{0,1\}^* \to \{1, \ldots, 2^{160}\}$ verwenden (siehe [MOV97, S. 349ff.]). Den Hashwert interpretiere man als binäre Darstellung einer Dezimalzahl

und addiere 1 hinzu. Damit die Sicherheitsaussage aus Satz 3.3.11 anwendbar ist (siehe oben), kann man den Exponenten k zufällig aus $\{1, \ldots, 2^{400}\}$ wählen.

3.3.5 Kryptoverfahren der Kategorie C

Wir beschreiben nun Kryptoverfahren, die von uns vollkommen neu entwickelt worden sind.

3.3.5.1 Signaturverfahren R-Feige-Fiat-Shamir (R-FFS)

In diesem Abschnitt beschreiben wir das R-Feige-Fiat-Shamir-Signaturverfahren (R-FFS). Dieses entsteht aus dem R-Fiat-Shamir-Identifikationsverfahren (siehe Abschnitt 3.3.5.4), indem die Rolle des Verifizierers durch eine Einweg-Hashfunktion ersetzt wird: $e = h(M\|\xi)$, wobei M den zu signierenden Text und ξ den Zeugen repräsentiert. Mit dieser in [FS87] vorgestellten Technik lässt sich jedes dreistufige (interaktive) Identifikationsverfahren (Zeuge-Challenge-Response) in ein (nicht-interaktives) Signaturverfahren überführen. Durch die Einwegeigenschaft der Hashfunktion bleibt die Pseudozufälligkeit der Challenge gewahrt.

Das Protokoll

Die Zahlen $k \in \mathbb{Z}_{\geqslant 2}$ und $m, t \in \mathbb{Z}_{\geqslant 0}$ werden fest für alle Teilnehmer vorgegeben. Allen Teilnehmern wird die kryptographische Hahsfunktion $h : \{0,1\}^* \to \{0,1\}^m$ bekannt gegeben. Aus technischen Gründen legen wir bei der Konstruktion des Signaturverfahrens folgende Variante des Fiat-Shamir-Identifikationsverfahrens zugrunde: Ein Teilnehmer **A** verwendet die privaten Schlüssel σ_j und die öffentlichen Schlüssel $\alpha_j = \sigma_j^{-k}$ (anstatt $\alpha_j = \sigma_j^k$) für $j \in \{1, \ldots, m\}$. Zur Verifikation prüft ein Verifizierer **B** die Gültigkeit der Gleichung $\xi = \eta^k \cdot \prod_{j=1}^m \alpha_j^{e_j}$.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle eine endliche Abelsche Gruppe G;
wähle zufällige Elemente $\sigma_j \in G$ und berechne $\alpha_j = \sigma_j^{-k}$, $j = 1, \ldots, m$.

As öffentlicher Schlüssel ist $(G, \alpha_1, \ldots, \alpha_m)$, der private Schlüssel ist $(\sigma_1, \ldots, \sigma_m)$.

Signatur: **A** führt die folgenden Schritte aus:
Wähle ein zufälliges Element $\varrho \in G$;
berechne $\xi = \varrho^k$;
berechne $e = (e_1, \ldots, e_m) = h(M\|\xi)$ $(e_i \in \{0,1\})$;
berechne $\eta = \varrho \cdot \prod_{j=1}^m \sigma_j^{e_j}$.

Die Signatur des Textes M ist $Sig_{\mathbf{A},M} = (e, \eta)$.

Verifikation: **B** berechnet $\xi' = \eta^k \cdot \prod_{j=1}^m \alpha_j^{e_j}$;
B berechnet $e' = (e'_1, \ldots, e'_m) = h(M\|\xi')$ $(e'_i \in \{0,1\})$;
B akzeptiert die Signatur $Sig_{\mathbf{A},M}$ genau dann, wenn $e = e'$ ist.

Das R-FFS-Signaturverfahren ist abgeschlossen *(complete)*, denn gültige Signaturen werden durch Teilnehmer, die dem Protokoll folgen, immer akzeptiert.

Proposition 3.3.12
Wenn $\boldsymbol{A}$ *und* $\boldsymbol{B}$ *dem vorstehenden R-FFS-Protokoll folgen, dann ist die Verifikation immer erfolgreich.*

Beweis: Mit den Bezeichnungen von oben gilt

$$\xi' = \eta^k \cdot \prod_{j=1}^{m} \alpha_j^{e_j} = \varrho^k \cdot \prod_{j=1}^{m} \sigma_j^{e_j k} \cdot \prod_{j=1}^{m} \sigma_j^{-e_j k} = \xi \ , \tag{3.8}$$

und daher ist $(e'_1, \ldots, e'_m) = h(M\|\xi') = h(M\|\xi) = (e_1, \ldots, e_m)$. □

Sicherheit

Die Sicherheit des R-FFS-Signaturverfahrens basiert auf dem Problem, k-te Wurzeln in der zugrunde liegenden Gruppe zu berechnen. Wer k-te Wurzeln berechnen kann, kann Signaturen für beliebige Texte fälschen, indem er aus dem öffentlichen Schlüssel eines Teilnehmers direkt dessen privaten Schlüssel berechnet.

Nach heutigem Kenntnisstand ist es in der Praxis unmöglich, k-te Wurzeln in Gruppen zu berechnen, die den Bedingungen aus Abschnitt 4.2 genügen, also insbesondere die praktische Berechnung der Gruppenordnung nicht zulassen. Ein effizienterer Angriff als k-te Wurzeln zu berechnen, ist uns nicht bekannt. Gegenstand künftiger Forschung bleibt die Untersuchung der Frage, ob das R-FFS-Signaturverfahren sicher im Sinne von Abschnitt 3.2.1 ist.

Effizienz

Der Signierer hat ein Gruppenelement zufällig zu wählen und eine k-te Potenz zu berechnen. (Letzteres erfordert $\leqslant 2 \cdot \lfloor \log k \rfloor - 1$ viele Gruppenoperationen). Dies kann komplett als Vorberechnung durchgeführt werden. Zudem hat der Signierer durchschnittlich $\frac{m}{2}$ viele Gruppenoperationen (im schlechtesten Fall m viele Gruppenoperationen) auszuführen.

Der Verifizierer muss eine k-te Potenz berechnen (keine Vorberechnung möglich) und durchschnittlich zudem $\frac{m}{2}$ viele Gruppenoperationen (im schlechtesten Fall m viele Gruppenoperationen) durchführen. Für die unten empfohlenen Parameter $m = 80$ und $k \in \{2, 3\}$ ergibt sich ein durchschnittlicher Aufwand von 41 $(k = 2)$ bzw. 42 $(k = 3)$ Gruppenoperationen sowohl für Signierer als auch für Verifizierer. Der Signierer muss zudem ein zufälliges Gruppenelement wählen, kann sich aber bei Vorabauswahl zufälliger Gruppenelemente durch Bilden der k-ten Potenz $\xi = \varrho^k$ eine (im Fall $k = 2$) bzw. zwei (im Fall $k = 3$) Gruppenoperationen sparen.

Empfehlung zur Parameterwahl

Im Gegensatz zum interaktiven R-FS-Identifikationsverfahren kann ein Angreifer im nichtinteraktiven R-FFS-Signaturverfahren vollkommen selbständig prüfen, ob ein Angriff mit den selbst gewählten Parametern erfolgreich sein wird. Um diese Offline-Attacken auszuschließen, empfehlen wir für das R-FFS-Signaturverfahren, den Sicherheitsparamter m größer zu wählen als den Sicherheitsparameter mt im R-FS-Identifikationsverfahren. Wir empfehlen die Wahl von $m = 80$. (Diese Empfehlung deckt sich mit den schärfsten Sicherheitsanforderungen aus [MOV97, S. 419].)

Zudem empfehlen wir aus Effizienzgründen die Verwendung von $k \in \{2, 3\}$, so dass die Sicherheit des Signaturverfahrens auf der Schwierigkeit basiert, Quadratwurzeln bzw. Kubikwurzeln zu berechnen.

Es sei bemerkt, dass die Berechnung von Quadratwurzeln im Spezialfall imaginär-quadratischer Zahlkörper – und nur da – polynomiell äquivalent zum Faktorisieren ganzer Zahlen (genauer: der Diskriminante des Zahlkörpers) ist. (Ein ausführlicher Beweis findet sich in [Mey97].)

3.3.5.2 Nichtabstreitbare Signaturen: CU-RDSA, DC-Schnorr, DC-GQ

Nichtabstreitbare Signaturen und Erweiterungen Nichtabstreitbare Signaturen (undeniable signatures) wurden in [Cv90] eingeführt. In einigen Anwendungen sind nichtabstreitbare Signaturen gewöhnlichen Signaturen vorzuziehen, weil nichtabstreitbare Signaturen nicht direkt verifiziert werden können, sondern nur über ein Verifikationsprotokoll, für dessen Durchführung der Verifizierer mit dem Signierer kooperieren muss. Der Signierer darf nicht abstreiten, eine Signatur ausgestellt zu haben, indem er nicht am Verifikationsprotokoll teilnimmt. Statt dessen gibt es in nichtabstreitbaren Signaturverfahren zudem ein Protokoll zur Ablehnung ungültiger Signaturen. Eine formale Definition, wann ein Signaturverfahren nichtabstreitbar heißt, findet sich in [BCDP91].

Eine Softwarefirma möchte beispielsweise zertifizieren, dass sie ein bestimmtes Programm geschrieben und an einen Kunden ausgeliefert hat. Daher versieht sie das Programm mit einer nichtabstreitbaren Signatur. Nur Kunden, die die Software direkt bei dieser Softwarefirma gekauft haben, können dann die Signatur verifizieren und sicher sein, dass nachträglich keine Viren in die Software eingeschleust wurden. Wer die Software illegal erworben hat, kann sich davon nicht überzeugen. Wenn die von der Firma verkaufte Software nicht das hält, was sie verspricht und gravierende Fehler enthält, sollte die Firma nicht in der Lage sein, ihre Signaturen anschließend abzustreiten. Dies sollte sie hingegen tun können, wenn ihre Software nachträglich durch einen Virus manipuliert wurde.

Zusätzlich zu den oben geschilderten Eigenschaften von nichtabstreitbaren Signaturen könnte es sinnvoll sein, dass der Signierer durch Veröffentlichung eines Teilgeheimnisses alle zuvor ausgestellten nichtabstreitbaren Signaturen in gewöhnliche Signaturen verwandeln kann. Dann könnten die Signaturen ohne Kooperation mit dem Signierer verifiziert werden, während Signaturen weiterhin in der Praxis nicht gefälscht werden könnten. Solche Signaturen heißen *konvertierbare nichtabstreitbare Signaturen (convertible undeniable signatures)*.

Diese wurden in [BCDP91] eingeführt.

In obigem Beispiel entstünde der Softwarefirma bei konvertierbaren nichtabstreitbaren Signaturen die Möglichkeit, das Teilgeheimnis, welches das Verifizieren oder Abstreiten einer Signatur erlaubt, mehreren Mitarbeitern der Firma anzuvertrauen. Dadurch könnten diese fortan im Namen der Softwarefirma mit Kunden kooperieren, um Signaturen zu verifizieren oder abzustreiten. Es wäre den Mitarbeitern aber nicht möglich, neue Signaturen im Namen der Firma zu erstellen. Im Falle einer gewöhnlichen nichtabstreitbaren Signatur hätte die Firma den entsprechenden Mitarbeitern aber das komplette Geheimnis verraten müssen, so dass die Mitarbeiter auch neue Signaturen für die Firma ausstellen könnten, was ein größeres Sicherheitsrisiko darstellen würde. Ein weiterer Vorteil entsteht, wenn die Softwarefirma Konkurs anmeldet. Dann könnte sie das Teilgeheimnis, welches zum Verifizieren von Signaturen befähigt, veröffentlichen. Dadurch könnte der Kunde sich

sicher sein, dass seine Softwareversion tatsächlich von der Firma stammt und nicht durch einen Virus manipuliert wurde.

Nichtabstreitbare Signaturen erscheinen auch in Situationen sinnvoll, in denen eine Person ein Dokument signieren möchte, ohne dass die Öffentlichkeit (z.B. Journalisten) zunächst verifizieren kann, dass die Signatur von dieser Person stammt. Stellen wir uns weiterhin vor, obige Person müsse irgendeine Partei dazu befähigen, den Gültigkeitsnachweis für von der Person ausgestellte Signaturen zu führen und für andere Signaturen deren Ungültigkeit nachzuweisen, ohne die Partei dadurch in die Lage zu versetzen, Signaturen zu fälschen. Ein Beispiel einer solchen Situation ist ein elektronisch verfasstes Testament. Dessen Inhalt möchte der Signierer zu Lebzeiten nicht veröffentlichen. Nach dem Tod des Signierers muss dessen Notar die Gültigkeit des Testamentes nachweisen können. Der Notar selbst darf dabei nicht in der Lage sein, das Testament zu manipulieren und für die manipulierte Version eine gültige Signatur zu erstellen. Der Signierer könnte hierzu sein Testament mit einer konvertierbaren nichtabstreitbaren Signatur versehen und diese Signatur zu einer gewöhnlichen Signatur konvertieren, indem er seinem Notar das Teilgeheimnis verrät, mit dessen Kenntnis der Notar künftig nachweisen kann, dass Signaturen gültig oder ungültig sind. In diesem Beispiel wäre es für den Signierer wünschenswert, wenn nicht alle von ihm vorher ausgestellten Signaturen durch Weitergabe des Teilgeheimnisses in gewöhnliche Signaturen konvertiert würden, so dass der Notar sie verifizieren kann. Statt dessen soll nur die Signatur, die zum Testament gehört, in eine gewöhnliche Signatur konvertiert werden. Verfahren, die neben dem Konvertieren aller Signaturen auch das Konvertieren einzelner ausgewählter Signaturen in gewöhnliche Signaturen ermöglichen, heißen *selektiv konvertierbare nichtabstreitbare Signaturverfahren (selectively convertible undeniable signatures)*. Diese wurden ebenfalls in [BCDP91] eingeführt.

Die bislang publizierten konvertierbaren nichtabstreitbaren Signaturverfahren setzen alle die Kenntnis der Ordnung der zugrunde liegenden Gruppe voraus. Die in der Literatur diesbezüglich geführten Sicherheitsbeweise nutzen meist aus, dass die Gruppenordnung prim ist. Die Ordnung von Klassengruppen algebraischer Zahlkörper ist aber im Allgemeinen nicht prim; sie kann (ebensowenig wie ein Vielfaches) in der Praxis nicht bestimmt werden. Daher war zunächst vollkommen unklar, wie man konvertierbare nichtabstreitbare Signaturen in Klassengruppen algebraischer Zahlkörper konstruieren kann. Wir stellen weiter unten ein Verfahren vor, mit dem man selektiv konvertierbare nichtabstreitbare Signaturen in Klassengruppen algebraischer Zahlkörper erstellen kann und führen einen Sicherheitsbeweis.

Ein Nachteil von nichtabstreitbaren Signaturen (und insbesondere auch von konvertierbaren nichtabstreitbaren Signaturen) liegt darin, dass der Signierer nicht erreichbar sein könnte oder es ablehnen könnte zu kooperieren. Fehlt es dem Signierer an Kooperationsbereitschaft, wird dieser Nachteil auch nicht durch konvertierbare nichtabstreitbare Signaturen behoben. Dann ist die Verifikation einer mutmaßlich von ihm ausgestellten Signatur nicht möglich. Chaum [Cha95] führte daher *Signaturen mit designiertem Beglaubiger (designated confirmer signatures)* ein. In diesen Verfahren befähigt ein Signierer a priori eine Person oder Instanz, an der Stelle des Signierers nachzuweisen, dass eine Signatur gültig ist. Ist der Signierer später nicht erreichbar oder verweigert seine Kooperation, tritt die vorher bestimmte Person an seine Stelle, um beim Nachweis der Gültigkeit oder Ungültigkeit einer vorgelegten Signatur mitzuwirken. Eine formale Definition eines Signaturverfahrens, mit dem man Signaturen mit designiertem Beglaubiger erzeugen kann, findet sich in [Oka94].

Auch hier gilt das oben Gesagte: Die bislang vorgestellten Signaturverfahren mit designiertem Beglaubiger setzen alle voraus, dass die zugrunde liegende Gruppe Primzahlordnung hat und

die Ordnung bekannt ist. Da diese Situation in Klassengruppen algebraischer Zahlkörper nicht gegeben ist, war zunächst keineswegs klar, wie man hier Signaturen mit designiertem Beglaubiger konstruieren soll. Wir stellen weiter unten ein Verfahren vor, mit dem man Signaturen mit designiertem Beglaubiger in Klassengruppen algebraischer Zahlkörper erstellen kann.

Alle im Folgenden zur Durchführbarkeit und Sicherheit der Verfahren getroffenen Aussagen gelten in beliebigen endlichen Abelschen Gruppen, die den Bedingungen aus Abschnitt 4.2 genügen.

CU-RDSA – Konvertierbare nichtabstreitbare Signaturen

Wir stellen nun ein konvertierbares nichtabstreitbares Signaturverfahren (convertible undeniable signature scheme) vor, welches auf dem RDSA-Signaturverfahren basiert. Das neue Verfahren nennen wir CU-RDSA (Convertible Undeniable - Root Based Digital Signature Algorithm). Das Verfahren ist anwendbar in endlichen Abelschen Gruppen mit unbekannter Gruppenordnung, insbesondere also in Klassengruppen algebraischer Zahlkörper.

Wir orientieren uns an dem in [BCDP91] vorgestellten Verfahren für konvertierbare nichtabstreitbare Signaturen. Dessen Anwendung in Klassengruppen algebraischer Zahlkörper ist aus folgenden Gründen unmöglich:

a) Das Protokoll [BCDP91] basiert auf dem ElGamal-Signaturverfahren in einer zyklischen Gruppe mit bekannter Primzahlordnung. In unserem Fall ist die Grupenordnung aber unbekannt und im Allgemeinen nicht prim.

b) Das Protokoll [BCDP91] arbeitet in einer Untergruppe der primen Restklassengruppe modulo einer Primzahl. Daher können Gruppenelemente als ganze Zahlen interpretiert und somit im Protokoll als Exponenten verwendet werden.

Wir modifizieren das Verfahren [BCDP91] folgendermaßen zu CU-RDSA, um ein in Klassengruppen algebraischer Zahlkörper anwendbares Protokoll zu erhalten:

a) Wie beim Design des RDSA-Protokolls (siehe Abschnitt 3.3.4.1) ersetzen wir die prime Untergruppenordnung im Protokoll [BCDP91] durch eine beliebige, aber fixe große Primzahl q und reduzieren die im Protokoll vorkommenden Exponenten modulo q. Unser Protokoll CU-RDSA basiert also auf RDSA, wohingegen das Protokoll [BCDP91] auf dem ElGamal-Signaturverfahren basiert.

b) Wir führen eine Hashfunktion f ein, die Gruppenelemente auf natürliche Zahlen abbildet.

Mit diesen massiven Protokolländerungen ist a priori keineswegs klar, ob die Sicherheitsaussagen aus [BCDP91] auch für CU-RDSA gelten. Daher diskutieren wir weiter unten die Sicherheit von CU-RDSA explizit. Im Vergleich zu [BCDP91] erhalten wir dabei tatsächlich analoge Sicherheitsaussagen für CU-RDSA.

Im Folgenden präsentieren wir zunächst Schlüsselerzeugung, Signaturerstellung und Verifikation einer CU-RDSA-Signatur. Wir treffen Aussagen zur Abgeschlossenheit und Sicherheit des Verfahrens. Dann präsentieren wir das Protokoll zur Ablehnung falscher Signaturen, zeigen dessen Abgeschlossenheit und Sicherheit. Anschließend erklären wir, wie der Signierer alle bislang erstellten Signaturen bzw. einzelne ausgewählte Signaturen in gewöhnliche Signaturen konvertiert. Abschließend diskutieren wir die Sicherheit des CU-RDSA-Signaturverfahrens.

Sei G eine endliche Abelsche Gruppe und γ ein zufällig gewähltes Gruppenelement. Sei q eine zufällig gewählte Primzahl mit $2^{\tilde{t}-1} < q \leqslant 2^{\tilde{t}}$ für einen Sicherheitsparameter $\tilde{t} \in \mathbb{Z}_{>0}$, z.B. $\tilde{t} = 160$. Sei $f : G \to \{1, \ldots, q-1\}$ eine kryptographische Hashfunktion. Der private Schlüssel eines Teilnehmers **A** sei eine Zufallszahl $a \in \{2, \ldots, q-1\}$, der öffentliche Schlüssel (G, γ, α, q) mit $\alpha = \gamma^a$. **A**s RDSA-Signatur für einen Text $M \in \{1, \ldots, q-1\}$ ist ein Paar (s, ϱ, λ) mit $\gamma^s \alpha^M \lambda^q = \varrho$. **A** bildet die Signatur durch Wahl von $k \in \{1, \ldots, q-1\}$ und Berechnen von $\varrho, x, l, s, \lambda$ mit $\varrho = \gamma^k$, $x = -a \cdot M + k$, $x = lq + s$ $(0 \leqslant s < q)$, $\lambda = \gamma^l$.

Die konvertierbare nichtabstreitbare Signatur für den Text M ist $Sig_{\mathbf{A},M}^{CU-RDSA} = (\alpha^t, \gamma^t, s, \varrho, \lambda)$, wobei $t \in \{1, \ldots, q-1\}$ zufällig gewählt und (s, ϱ, λ) die RDSA-Signatur für $\tilde{M} = t \cdot f(\gamma^t) \cdot h(M \| \varrho) \cdot z$ ist. Hierbei ist z eine geheime Zahl, die **A** zufällig in $\{2, \ldots, q-1\}$ gewählt hatte, und es sei $\zeta = \alpha^z$. Beim RDSA-Signaturverfahren (wie bei allen ElGamal-ähnlichen Signaturverfahren) kann ein Angreifer existentielle Fälschungen (s', ϱ', λ') für Texte M' erstellen, falls auf den Einsatz einer Hashfunktion verzichtet wird. Allerdings kann bei diesem Angriff M' nicht beliebig gewählt werden, so dass M' in der Regel kein sinnvoller Text ist. Dem tragen wir Rechnung, indem wir beim neuen Verfahren CU-RDSA die kryptographische Hashfunktion $h : \{0,1\}^* \to \{0, \ldots, q-1\}$ verwenden. Das Quintupel $(T_1, T_2, s, \varrho, \lambda)$ ist eine gültige Signatur für den Text M genau dann, wenn

$$\left(T_1^{f(T_2)h(M\|\varrho)}\right)^z = \gamma^{-s} \cdot \varrho \cdot \lambda^{-q} .$$

In diesem Abschnitt bezeichnen wir den Ausdruck $\gamma^{-s}\varrho\lambda^{-q}$ mit v und $T_1^{f(T_2)h(M\|\varrho)}$ mit ω. Das Verifizieren einer Signatur $(T_1, T_2, s, \varrho, \lambda)$ für M ist also gleichbedeutend damit zu entscheiden, ob $\log_\omega v = \log_\alpha \zeta$.

CU-RDSA – Das Verfahren

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:

Wähle eine endliche Abelsche Gruppe G und eine hinreichend große zufällige Primzahl q mit $2^{\tilde{t}-1} < q \leqslant 2^{\tilde{t}}$ für einen Sicherheitsparameter $\tilde{t} \in \mathbb{Z}_{>0}$ (z.B. $\tilde{t} = 160$);
wähle ein zufälliges Element $\gamma \in G$;
wähle Zufallszahlen a, z aus $\{2, \ldots, q-1\}$ und berechne $\alpha = \gamma^a, \zeta = \alpha^z$.

As öffentlicher Schlüssel ist $K_P = (G, \gamma, \alpha, \zeta, q)$, **A**s private Schlüssel sind $K_{S_1} = a$ und $K_{S_2} = z$.

Signatur: **A** führt die folgenden Schritte aus:

Wähle Zufallszahlen t, k aus $\{0, \ldots, q-1\}$;
berechne $T_1 = \alpha^t$, $T_2 = \gamma^t$, $\varrho = \gamma^k$;
berechne $x = -a\tilde{M} + k$ für $\tilde{M} = tf(\gamma^t)h(M\|\varrho)z$;
berechne ganze Zahlen s und l, so dass $x = lq + s$ und $0 \leqslant s < q$;
berechne $\lambda = \gamma^l$.

Die nichtabstreitbare konvertierbare Signatur des Textes M ist $Sig_{\mathbf{A},M}^{CU-RDSA} = (T_1, T_2, s, \varrho, \lambda)$.

Verifikation: **A** (der Signierer) und **B** (der Verifizierer) berechnen $\omega = T_1^{f(T_2)h(M\|\varrho)}$ und $v = \gamma^{-s}\varrho\lambda^{-q}$. **A** führt den nachfolgend beschriebenen Zero-Knowledge-Beweis, dass $\log_\omega v =$

$\log_\alpha \zeta$ ist.
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Beweis, dass $\log_\omega v = \log_\alpha \zeta$. A kennt $z = \log_\alpha \zeta$. ([CEvdG88])
Zu Beginn teilt **A** der Partei **B** die Eingabewerte mit: $G, \omega, v, \alpha, \zeta, L$, wobei L die Größenordnung der Gruppenordnung angibt: $L \approx |G|, L \geqslant |G|$. Die folgenden Schritte werden ν mal wiederholt.

(1) **A** wählt ein zufälliges $r \in \{1, \ldots, L\}$;
A berechnet $\gamma_1 = \omega^r$, $\gamma_2 = \alpha^r$;
A sendet γ_1, γ_2 an **B**.

(2) **B** wählt ein Zufallsbit $b \in \{0, 1\}$;
B sendet b an **A**.

(3) **A** berechnet $y' = r + bz$ und ganze Zahlen L', y mit $0 \leqslant y < L$ und $y' = L'L + y$;
A berechnet $\delta_1 = \omega^{L'}$, $\delta_2 = \alpha^{L'}$;
A sendet y, δ_1, δ_2 an **B**.

(4) **B** verifiziert, dass $\omega^y = \gamma_1 v^b \delta_1^{-L}$ und $\alpha^y = \gamma_2 \zeta^b \delta_2^{-L}$ gilt.

Hierin ist $\nu \in \mathbb{Z}_{>0}$ der Sicherheitsparameter, auf den sich beide Parteien vor der Protokollausführung geeinigt haben. Wir nehmen an, dass ν durch ein Polynom in $\log\ L$ beschränkt ist. Mit wachsendem ν reduzieren sich **A**s Chancen eines unbemerkten Betruges exponentiell, während der Kommunikations- und Berechnungsaufwand nur linear steigt.

Satz 3.3.13
*(a) Wenn **B** sich an das Beweisprotokoll hält und **A** den diskreten Logarithmus z nicht kennt, dann bemerkt **B** jeden Betrugsversuch von **A** mit einer Wahrscheinlichkeit $\geqslant 1 - 2^{-\nu}$.*

*(b) Wenn **A** sich an das Beweisprotokoll hält, dann existiert für jeden vorstellbaren Polynomzeit-Verifizierer **B** ein Polynomzeit-Simulator für **A**.*

Beweis: Der Beweis erfolgt analog zu [CEvdG88, Beweis zu Theorem 4]. □

Lemma 3.3.14
*Wenn $(T_1, T_2, s, \rho, \lambda)$ eine von **A** korrekt ausgestellte CU-RDSA-Signatur für den Text M ist, gilt folgendes:*

- *Der Verifizierer **B** akzeptiert die Signatur mit Wahrscheinlichkeit 1.*
- *Das interaktive Wissensbeweisprotokoll zur Verifikation von CU-RDSA-Signaturen besitzt die Zero-Knowledge-Eigenschaft.*

*Wenn $(T_1, T_2, s, \rho, \lambda)$ keine gültige CU-RDSA-Signatur für den Text M ist, akzeptiert **B** mit vernachlässigbar kleiner Wahrscheinlichkeit, d.h. vernachlässigbar in der binären Länge von q.*

Beweis: Die Behauptungen ergeben sich unmittelbar aus Satz 3.3.13. Insbesondere lässt sich der in [CEvdG88, Beweis zu Theorem 4] beschriebene Polynomzeit-Simulator für **A**s Rolle im Beweisprotokoll leicht zu einem Polynomzeit-Simulator für **A**s Rolle im Protokoll zur Verifikation einer CU-RDSA-Signatur erweitern. Es bleibt lediglich zu zeigen, dass **B** obigen Zero-Knowledge-Beweis mit Wahrscheinlichkeit 1 akzeptiert, wenn **A** den diskreten Logarithmus $z = \log_\omega v = \log_\alpha \zeta$ weiß und sowohl **A** als auch **B** dem Beweisprotokoll folgen. In diesem Fall ergibt sich bei **B**s Prüfung im letzten Schritt des Beweisprotokolls: $\omega^y = \omega^{y'}(\omega^{L'})^{-L} = \omega^r(\omega^z)^b(\omega^{L'})^{-L} = \gamma_1 v^b \delta_1^{-L}$ und $\alpha^y = \alpha^{y'}(\alpha^{L'})^{-L} = \alpha^r(\alpha^z)^b(\alpha^{L'})^{-L} = \gamma_2 \zeta^b \delta_2^{-L}$. □

Wenn $(T_1, T_2, s, \rho, \lambda)$ keine gültige CU-RDSA-Signatur für den Text M ist, funktioniert die Simulation des Protokolls zur Verifikation einer CU-RDSA-Signatur dennoch. Dabei kann nicht ausgedrückt werden, dass es sich um eine Simulation mit einer gefälschten Signatur handelt, sofern man nicht ohnehin zwischen gültigen und ungültigen Signaturen unterscheiden kann. Daher kann man nach Aufzeichnen der beim Verifikationsprotokoll übermittelten Daten keinen Beweis führen, dass eine CU-RDSA-Signatur korrekt sei.

Protokoll zur Ablehnung falscher Signaturen

Um eine gefälschte Signatur abzulehnen, d.h. um zu zeigen, dass eine vorgegebene Signatur gefälscht wurde, führt **A** in Kooperation mit **B** folgendes Protokoll aus:

(1) **A** und **B** berechnen $\omega = T_1^{f(T_2)h(M\|\varrho)}$ und $v = \gamma^{-s} \cdot \varrho \cdot \lambda^{-q}$.

(2) **A** führt den nachfolgend beschriebenen Zero-Knowledge-Beweis, dass $\log_\omega v \neq \log_\alpha \zeta$ ist.

(3) **B** akzeptiert die Ablehnung der Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

A muss insbesondere beweisen, dass zwei diskrete Logarithmen unterschiedlich sind, ohne irgendetwas über diese diskreten Logarithmen preiszugeben. Wir stellen ein hierzu geeignetes interaktives Wissensbeweisprotokoll vor. Dabei bezeichne $BC(b, R)$ ein Bit Commitment (Blob) für das Bit b unter Benutzung des Zufallsstrings R. Der Blob wird geöffnet, indem R preisgegeben wird. Der Sicherheitsparameter $\nu \in \mathbb{Z}_{>0}$, auf den sich beide Parteien vor der Protokollausführung einigen, ist durch ein Polynom in $\log\ L$ beschränkt.

Beweis, dass $\log_\omega v \neq \log_\alpha \zeta$. A kennt $z = \log_\alpha \zeta$.

Zu Beginn teilt **A** der Partei **B** die Eingabewerte mit: $G, \omega, v, \alpha, \zeta, L$, wobei L die Größenordnung der Gruppenordnung angibt: $L \approx |G|, L \geqslant |G|$. Die folgenden Schritte werden ν mal wiederholt.

(1) **B** wählt eine zufällige Primzahl $e \in \{2, \ldots, L\}$;
B wählt ein Zufallsbit $b \in \{0, 1\}$;
B berechnet im Falle $b = 0$ die Werte $\delta_1 = \alpha^e$, $\delta_2 = \zeta^e$ und im Falle $b = 1$ die Werte $\delta_1 = \omega^e$, $\delta_2 = v^e$;
B sendet δ_1, δ_2 an **A**.

(2) Falls $\delta_1^z = \delta_2$, setzt **A** $b' = 0$; andernfalls setzt **A** $b' = 1$;
A wählt einen Zufallsstring R, setzt $blob = BC(b', R)$ und sendet $blob$ an **B**.

(3) **B** sendet e an **A**.

(4) Falls $(\delta_1, \delta_2) = (\alpha^e, \zeta^e)$ oder $(\delta_1, \delta_2) = (\omega^e, v^e)$ ist, setzt **A** $ans = R$; andernfalls setzt **A** $ans = stop$;
A sendet ans an **B**.

(5) Falls $ans = stop$, stoppt **B** die Ausführung des Protokolls; andernfalls verifiziert **B**, dass $BC(b, ans) = blob$ gilt.

Satz 3.3.15

*(a) Wenn $\log_\omega v \not\equiv \log_\alpha \zeta \ mod \ |<\omega>|$ gilt und **A** den Wert $z = \log_\alpha \zeta$ kennt und **A**, **B** sich an das Protokoll halten, dann akzeptiert **B** den Beweis mit Wahrscheinlichkeit 1.*

*(b) Wenn $\log_\omega v \equiv \log_\alpha \zeta \ mod \ | < \omega > |$ gilt, **B** sich an das Protokoll hält und **B** nicht probabilistisch in Polynomzeit entscheiden kann, ob ein vorgegebenes Gruppenelement κ in der von einem gegebenen Gruppenelement η erzeugten Untergruppe liegt, dann lehnt **B** den Beweis mit Wahrscheinlichkeit $\geqslant 1 - 2^{-t}$ ab.*

(c) Das obige interaktive Wissenbeweisprotokoll besitzt die Zero-Knowledge-Eigenschaft.

Beweis:

(a) Angenommen, es gelte $\log_\omega v \not\equiv \log_\alpha \zeta \ mod \ |<\omega>|$. Dann wird **A** immer $b' = b$ finden: Wenn $b = 0$ ist, dann ergibt sich $\delta_1^z = \alpha^{ez} = \zeta^e = \delta_2$, also $b' = 0$.
Wenn $b = 1$ ist, nehmen wir an, **A** fände $b' = 0$. Wegen $\delta_1^z = \delta_2$ ergäbe sich dann $\omega^{ez} = \delta_1^z = \delta_2 = v^e$. Da e mit sehr großer Wahrscheinlichkeit eine Primzahl in der Größenordnung der Gruppenordnung ist, können wir o.B.d.A. annehmen, dass e teilerfremd zur Gruppenordnung ist: $ggT(e, |G|) = 1 = x_1 e - x_2 |G|$ für geeignete ganze Zahlen x_1, x_2. Es würde folgen $(\omega^z)^{ex_1} = v^{ex_1}$, also $\omega^z \cdot (\omega^{zx_2})^{|G|} = v \cdot (v^{x_2})^{|G|}$. Demnach ergäbe sich $\omega^z = v$, also $\log_\omega v = \log_\alpha \zeta$, im Widerspruch zur Annahme. Daher findet **A** im Falle $b = 1$ auch $b' = 1$. Daher wird **B** den Beweis zur Ablehnung einer Signatur akzeptieren.

(b) Angenommen, es gelte $\log_\omega v \equiv \log_\alpha \zeta \ mod \ | < \omega > |$. **A**s Prüfung würde für jede Wahl des Bits b $\delta_1^z = \delta_2$ ergeben. Um zu prüfen, ob **B** (α^e, ζ^e) oder (ω^e, v^e) sendet, müsste **A** zumindest das Problem lösen, ob ein vorgegebenes Gruppenelement κ in der von einem gegebenen Element η erzeugten Untergruppe liegt. Dieses Entscheidungsproblem gilt gemeinhin als genauso schwer wie das diskrete Logarithmusproblem. Nach Voraussetzung kann **A** dieses Problem nicht probabilistisch in Polynomzeit lösen. Daher bleibt **A** keine bessere Möglichkeit als das Bit b zu raten. Bei ν Durchläufen ergibt sich für **A** demnach eine Wahrscheinlichkeit von $2^{-\nu}$, eine tatsächlich gültige Signatur als falsche Signatur abzulehnen.

(c) Das Beweisprotokoll kann mit den gleichen Techniken simuliert werden, die in [CEvdG88, Beweis zu Theorem 4] verwendet werden. Daher besitzt dieses interaktive Wissensbeweisprotokoll die Zero-Knowledge-Eigenschaft.

□

Lemma 3.3.16

1. *Wenn $(T_1, T_2, s, \rho, \lambda)$ eine falsche Signatur für den Text M ist, akzeptiert der Verifizierer **B** die Ablehnung der Signatur mit Wahrscheinlichkeit 1.*

2. *Wenn $(T_1, T_2, s, \rho, \lambda)$ eine korrekt ausgestellte CU-RDSA-Signatur für den Text M ist, lehnt der Verifizierer **B** die Ablehnung der Signatur mit hoher Wahrscheinlichkeit ab (d.h. er akzeptiert die Ablehnung mit in der binären Länge von q vernachlässigbar kleiner Wahrscheinlichkeit), unabhängig vom Verhalten des Signierers **A**.*

3. *Das interaktive Wissensbeweisprotokoll zur Ablehnung falscher CU-RDSA-Signaturen besitzt die Zero-Knowledge-Eigenschaft.*

Beweis: Der Beweis erfolgt analog zum Beweis von Lemma 3.3.14. □

Konvertierung aller Signaturen. Eine nichtabstreitbare Signatur kann durch Veröffentlichung des privaten Schlüssels K_{S_2} in eine gewöhnliche digitale Signatur konvertiert werden. Denn mit Kenntnis von $K_{S_2} = z$ und des öffentlichen Schlüssels K_P kann jedermann eine Signatur $(T_1, T_2, s, \rho, \lambda)$ für einen Text M verifizieren. Hierzu muss lediglich überprüft werden, ob $\left(T_1^{f(T_2)h(M\|\varrho)}\right)^z = \gamma^{-s}\varrho\lambda^{-q}$ gilt. Nach Veröffentlichung von $K_{S_2} = z$ sind alle bislang und künftig von **A** ausgestellten Signaturen äquivalent zu RDSA-Signaturen. Jeder, der $K_{S_2} = z$ weiß, hätte die Rolle des Beweisers (Provers) in den vorausgegangenen Ausführungen der Protokolle zum Verifizieren und Ablehnen von CU-RDSA-Signaturen einnehmen können. Die dabei über einen unsicheren Kanal gesendeten Daten helfen einem Angreifer daher nicht, nachdem $K_{S_2} = z$ veröffentlicht wurde.

Konvertierung ausgewählter Signaturen. Eine ausgewählte Signatur $(T_1, T_2, s, \rho, \lambda)$ für einen Text M kann in eine gewöhnliche digitale Signatur konvertiert werden, indem der bei der Signaturerstellung verwendete Exponent t veröffentlicht wird. Es gilt $\omega^z = \left(T_1^{f(T_2)h(M\|\varrho)}\right)^z = (\alpha^z)^{tf(T_2)h(M\|\varrho)} = \zeta^{tf(T_2)h(M\|\varrho)}$. Daher kann jeder, der t weiß, folgendermaßen feststellen, ob $(T_1, T_2, s, \rho, \lambda)$ eine gültige CU-RDSA-Signatur für den Text M ist: Prüfe, ob $T_1 = \alpha^t$ und $T_2 = \gamma^t$ und $\zeta^{tf(T_2)h(M\|\varrho)} = \gamma^{-s}\varrho\lambda^{-q}$ gilt. Das Konvertieren einer ausgewählten Signatur setzt also voraus, dass sich der Signierer **A** jeweils an den zum Signieren der Texte verwendeten zufällig gewählten Exponenten t erinnert. Deswegen empfehlen wir, in der Praxis einmalig einen geheimen Schlüssel $\tilde{t}$ für eine Pseudozufallsfunktion $f_{\tilde{t}}$ zu wählen (siehe [GGM84]) und dann $t = f_{\tilde{t}}(M)$ für einen zu signierenden Text M zu wählen. Die Eigenschaften von Pseudozufallsfunktionen garantieren, dass es einem Angreifer unmöglich ist, $f_{\tilde{t}}(M)$ für einen Text M zu finden, selbst wenn polynomiell viele Paare $(M_i, f_{\tilde{t}}(M_i))$ gegeben sind ($M \neq M_i$ für alle i). Daher bleibt nach Konvertierung polynomiell vieler Signaturen (polynomiell in der binären Länge von q) die Nichtabstreitbarkeit anderer Signaturen erhalten.

Sicherheit. Wir diskutieren zunächst die Möglichkeit, Signaturen zu fälschen. Die interaktiven Wissenbeweisprotokolle zur Verifikation und Ablehnung von CU-RDSA-Signaturen besitzen die Zero-Knowledge-Eigenschaft. Also erhält ein Angreifer durch die Beobachtung von Verifikationen oder Ablehnungen von CU-RDSA-Signaturen keinerlei verwertbare Informationen. Der stärkste vorstellbare Angriff ist daher ein angepasst-gewählter Text-Angriff (adaptive-chosen message attack): Der Angreifer **E** erhält auf irgendeine Weise Kenntnis von gültigen Signaturen für Texte seiner Wahl und versucht anschließend, für einen anderen Text eine gültige Signatur zu erstellen. Der Angreifer **E** kennt also Quintupel $(T_1, T_2, s, \rho, \lambda)$ mit $T_1^{f(T_2)h(M\|\varrho)z} = \gamma^{-s}\varrho\lambda^{-q}$, wobei

M jeweils durch den Angreifer **E** und T_1, T_2 durch den Signierer **A** gewählt wurden. Im Folgenden nehmen wir den ungünstigsten Fall an, dass **E** jeweils die von **A** benutzten Exponenten $t = \log_\alpha T_1 = \log_\gamma T_2$ kennt. Das bedeutet, **E** erhält RDSA-Signaturen für Zahlen der Form $tf(T_2)h(M\|\varrho)z$. Die einzige Konsequenz ist, dass **E** nun neue RDSA-signierte Texte konstruieren kann, wobei **E** diese neuen Texte allerdings nicht selbst wählen kann; statt dessen ergeben sich die neuen Texte durch Anwendung einer in der Praxis nicht invertierbaren Transformation der vorhandenen signierten Texte (siehe [ElG85]) und sind für **E** damit nicht zu steuern. Allerdings kann **E** auch ohne Kenntnis gültiger Signaturen einen RDSA-signierten Text erzeugen. Auch hier hat **E** in der Praxis keine Kontrolle über den resultierenden signierten Text M. Entsprechend kann man für alle ElGamal-ähnlichen Signaturverfahren neue signierte Texte erzeugen, über die man keine Kontrolle hat (siehe Abschnitt 3.3.4.1 und [ElG85]). Dieses Problem kann durch Anwenden einer kryptographischen Hashfunktion auf den Text vor dem Signieren vermieden werden.

Nehmen wir nun an, der Angreifer **E** könne in unserem Verfahren irgendeine gültige Signatur $(T_1, T_2, s, \varrho, \lambda)$ für einen Text M erstellen. Wir unterscheiden zwei Fälle:

Fall 1: Der Angreifer **E** kennt die Zahl t mit $T_1 = \alpha^t, T_2 = \gamma^t$.
Dann hat **E** eine RDSA-Signatur für den Text $tf(T_2)h(M\|\varrho)z$ erzeugt. Also hat **E** entweder einen vollkommen neuen Weg gefunden, RDSA-Signaturen zu fälschen, oder **E** kennt eine Methode, das Resultat der oben erwähnten in der Praxis nicht invertierbaren Funktion in der Form $tf(T_2)h(M\|\varrho)z$ zu schreiben. Nach unserem Kenntnisstand ist dies ein schwieriges Berechnungsproblem, für welches keine effiziente Lösung bekannt ist; denn f und h sind als kryptographische Hashfunktionen in der Praxis nicht invertierbar, und z ist eine Konstante. Daher kann bis auf t keiner der Faktoren $tf(T_2)h(M\|\varrho)z$ durch den Angreifer **E** einfach gesteuert werden.

Fall 2: Der Angreifer **E** kennt die Zahl $t = \log_\alpha T_1 = \log_\gamma T_2$ nicht.
Dennoch kann **E** Werte finden, die $T_1^{f(T_2)h(M\|\varrho)z} = \gamma^{-s}\varrho\lambda^{-q}$ erfüllen. Dieses Berechnungsproblem kann nur dann einfacher als das Berechnungsproblem aus Fall 1 sein, wenn **E** die Werte T_1 und $f(T_2)$ auf irgendeine Weise durch Kenntnis gültiger Signaturen, die er zuvor schon erhalten hatte, berechnen kann. Wie oben bereits diskutiert sind diese Signaturen RDSA-Signaturen spezieller Gestalt. Allerdings kann **E** diese Texte nicht nach eigenen Wünschen wählen, da sie einen Faktor der Form $tf(\gamma^t)$ enthalten, wobei t unabhängig und zufällig durch den Signierer **A** gewählt wird. Deshalb können wir annehmen, dass diese Signaturen dem Angreifer nichts nutzen. Daher könnte der Angreifer **E** genauso Werte T_1 und T_2 wählen, deren diskreten Logarithmus t er kennt. Damit sind wir wieder in Fall 1.

Wir diskutieren nun die Schwierigkeit des Problems, Signaturen ohne die Hilfe des Signierers zu verifizieren bzw. abzulehnen. Wie oben bewiesen besitzen die interaktiven Wissenbeweisprotokolle zur Verifikation und Ablehnung von CU-RDSA-Signaturen die Zero-Knowledge-Eiegenschaft. Daher nehmen wir an, dem Angreifer **E** lägen einige Signaturen vor, von deren Gültigkeit er weiß, und **E** versuche nun zu entscheiden, ob ein gegebenes Quintupel $(T_1, T_2, s, \varrho, \lambda)$ die gültige CU-RDSA-Signatur für einen Text M sei, d.h. ob die Gleichung $T_1^{f(T_2)h(M\|\varrho)z} = \gamma^{-s}\varrho\lambda^{-q}$ erfüllt sei, wobei **E** die Exponenten z und $t = \log_\alpha T_1 = \log_\gamma T_2$ nicht kenne. Der einzige uns bekannte Weg, die Gültigkeit einer derartigen Gleichung zu überprüfen, besteht darin, die Ausdrücke auf beiden Seiten zu berechnen und zu vergleichen. Nehmen wir an, der Angreifer könnte die linke Seite berechnen. Der Wert $h(M\|\varrho)$ ist unabhängig von T_1, T_2, z. Wir können davon ausgehen, dass **E** die linke Seite für beliebige Gruppenelemente ϱ (die vom Signierer vorgegeben werden)

berechnen kann. Daher ist es plausibel, dass **E** auch $T_1^{f(T_2)z}$ berechnen kann. Die kryptographische Hashfunktion f „vernichtet" auf Grund seiner Einwegeigenschaft jeden Zusammenhang zwischen T_1 und $f(T_2)$ in dem Sinne, dass $f(T_2)$ als Zufallszahl in $\{1, \ldots, q-1\}$ angesehen werden kann, die der Signierer unabhängig von T_2 und T_1 wählt. Daher können wir davon ausgehen, dass **E** auch T_1^z berechnen kann. Das Berechnen von $T_1^z = \alpha^{tz}$ bei gegebenen Werten $T_1 = \alpha^t$ und $\zeta = \alpha^z$ und unbekannten Exponenten t, z ist genau das Lösen des Diffie-Hellman-Problems, welches wir als in der Praxis unlösbar einstufen. Wir folgern, dass ein Angreifer ohne Kooperation mit dem Signierer keine Signaturen überprüfen kann.

Wenn dem Angreifer **E** gültige CU-RDSA-Signaturen $(T_{1i}, T_{2i}, s_i, \rho_i, \lambda_i)$ für Texte M_i (vom gleichen Signierer **A**) vorliegen, kennt **E** auch $\alpha^{t_i f(\gamma^{t_i}) h(M_i \| \varrho_i) z}$, $\zeta = \alpha^z$ und die unabhängig voneinander gewählten zufälligen t_i. (Wir nehmen den ungünstigsten Fall an, dass der Signierer alle t_i veröffentlicht hat.) Der Angreifer könnte dann versuchen, für ein neu gewähltes zufälliges t den Wert von $\alpha^{tf(\gamma^t)z}$ zu bestimmen, um anschließend passende (s, ϱ, λ) herzuleiten. Doch die oben erwähnten Werte $\alpha^{t_i f(\gamma^{t_i}) h(M_i \| \varrho_i) z}$, α^z und t_i könnte der Angreifer mit derselben Wahrscheinlichkeitsverteilung auch selbst ohne jegliche Hilfe generieren. Daher ist es plausibel anzunehmen, dass einem Angreifer die Kenntnis gültiger CU-RDSA-Signaturen nicht helfen, um für einen neuen Text M erfolgreich eine CU-RDSA-Signatur zu fälschen.

Die formale Definition der Nichtabstreitbarkeit einer Signatur aus [BCDP91] verlangt einen Simulator für CU-RDSA-Signaturen. Auf Grund der soeben geschilderten Zusammenhänge vermuten wir, dass eine Maschine, die ein Quintupel $(T_1, T_2, s, \varrho, \lambda)$ ausgibt mit zufälligen Gruppenelementen $T_1, T_2, \varrho, \lambda$ und einer Zufallszahl s aus $\{0, \ldots, q-1\}$, einen Signatur-Simulator für CU-RDSA-Signaturen darstellt. Die exakte Analyse dieser Frage bleibt Gegenstand künftiger Forschung.

DC-Schnorr und DC-GQ – Konvertierbare Signaturen mit designiertem Beglaubiger

Wir stellen nun ein auf dem Wurzel- und DL-Problem basierendes Schema vor, welches mit gewissen 3-Wege-Identifikationsprotokollen (interaktive Identifikationsprotokolle, in denen 3 Nachrichten zwischen Beweiser und Verifizierer ausgetauscht werden) kombiniert werden kann. Durch diese Kombination ergibt sich ein konvertierbares Signaturverfahren mit designiertem Beglaubiger (designated confirmer signature scheme). Eine Implementierung in Klassengruppen algebraischer Zahlkörper ist bei Kombination mit allen 3-Wege-Identifikationsprotokollen möglich, welche die Kenntnis der Gruppenordnung nicht erfordern. Wir kombinieren das Schema beispielhaft mit zwei verschiedenen 3-Wege-Identifikationsprotokollen:

- Das DC-Schnorr-Signaturverfahren ergibt sich durch Kombination mit einer geeigneten Variante des Schnorr-Identifikationsprotokolls.
- Das DC-GQ-Signaturverfahren ergibt sich durch Kombination mit einer geeigneten Variante des Guillou-Quisquater-Identifikationsprotokolls.

Die resultierenden Verfahren sind anwendbar in endlichen Abelschen Gruppen mit unbekannter Gruppenordnung, insbesondere also in Klassengruppen algebraischer Zahlkörper. Wir orientieren uns beim Design unseres Schemas an der Arbeit [Oka94]. Das Schema aus [Oka94] ist in unserem Kontext nicht anwendbar, da es in einer zyklischen Untergruppe mit bekannter Primzahlordnung arbeitet, während wir in einer Gruppe mit unbekannter Ordnung arbeiten, die im Allgemeinen

nicht prim ist. Wie beim Design des RDSA-Protokolls (siehe Abschnitt 3.3.4.1) ersetzen wir die prime Untergruppenordnung im Protokoll [Oka94] durch eine beliebige, aber fixe große Primzahl q und reduzieren die im Protokoll vorkommenden Exponenten modulo q.

Im Folgenden erläutern wir zunächst für unser allgemeines Schema, dann für die beiden Signaturverfahren mit designiertem Beglaubiger, folgende Komponenten:

- Schlüsselerzeugung für Signierer und designierten Beglaubiger
- Signaturerstellung durch den Signierer
- Verifikation einer Signatur, wobei der Verifizierer mit dem Signierer kooperiert.
- Verifikation einer Signatur, wobei der Verifizierer mit dem eingangs bestimmten Beglaubiger kooperiert *(Beglaubigung)*.
- Konvertierung einer Signatur mit designiertem Beglaubiger in eine gwöhnliche Signatur

Das allgemeine Schema. Sei (A_1, A_2, D) ein 3-Wege-Identifikationsprotokoll, so dass sich Partei **A** (Beweiser, Prover) bei Partei **B** (Verifizierer) wie folgt authentifiziert:

1. **A** sendet $\xi_1 = A_1(w)$ an **B**, wobei w eine Zufallszahl oder ein zufälliges Gruppenelement ist.
2. **B** sendet **A** eine Zufallszahl e.
3. **A** sendet $(\xi_2, \xi_3) = A_2(w, e, s)$ an **B**.
   B verifiziert, ob $\xi_1 = D(e, \xi_2, \xi_3, \alpha)$.

Hierbei ist α **A**s öffentlicher Schlüssel, s ist **A**s geheimer Schlüssel. Wir verwenden im Folgenden eine kryptographische Hashfunktion $H : \{0,1\}^* \to \{1, \ldots, L-1\}$, wobei die positive ganze Zahl L die Gruppenordnung approximiere.

Schlüsselerzeugung: **A** wählt eine endliche Abelsche Gruppe G, ein zufälliges Element $\gamma \in G$ und eine Zahl L in der Größenordnung der Gruppenordnung: $L \approx |G|, L \geqslant |G|$;
A wählt für das 3-Wege-Identifikationsprotokoll (A_1, A_2, D) einen geheimen Schlüssel s und einen zugehörigen öffentlichen Schlüssel α.
Der von **A** vorab bestimmte Beglaubiger **C** (designated confirmer) wählt einen geheimen Schlüssel $u \in \{1, \ldots, L-1\}$ und den öffentlichen Schlüssel $\beta = \gamma^u$.

Signatur: **A** führt die folgenden Schritte aus:
Wähle eine Zufallszahl r aus $\{1, \ldots, L-1\}$ und eine Zufallszahl oder ein zufälliges Gruppenelement w (wird weiter unten spezifiziert);
berechne $\delta = \gamma^r$;
berechne $\xi_1 = A_1(w)$;
berechne $e = \beta^r \oplus H(M, \xi_1)$;
berechne $(\xi_2, \xi_3) = A_2(w, e, s)$;

Die Signatur des Textes M mit designiertem Beglaubiger **C** ist $Sig_{\mathbf{A},M,DC\ \mathbf{C}} = (\delta, e, \xi_2, \xi_3)$.

Verifikation (**A** kooperiert mit **B**):
A (der Signierer) und **B** (der Verifizierer) berechnen $\zeta = e \oplus H(M, D(e, \xi_2, \xi_3, \alpha))$.
A führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \delta = \log_\beta \zeta$ (siehe Abschnitt 3.3.5.2).
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Beglaubigung (**C** kooperiert mit **B** zum Gültigkeitsnachweis der Signatur):
C (der Beglaubiger) erhält M und die Signatur $(\delta, e, \xi_2, \xi_3)$ von **A** oder **B**.
C und **B** (der Verifizierer) berechnen $\zeta = e \oplus H(M, D(e, \xi_2, \xi_3, \alpha))$.
C führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \beta = \log_\delta \zeta$ (siehe Abschnitt 3.3.5.2).
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Konvertierung einer Signatur $(\delta, e, \xi_2, \xi_3)$ für den Text M durch **C**:
C berechnet $\zeta = e \oplus H(M, D(e, \xi_2, \xi_3, \alpha))$.
C führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \beta = \log_\delta \zeta$ (siehe Abschnitt 3.3.5.2).
C wählt $t \in \{1, \ldots, L-1\}$ zufällig.
C berechnet $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in G$ und $k', L', k \in \mathbb{Z}$ mit $0 \leqslant k < L$ und

$$\begin{aligned} \lambda_1 &= \gamma^t, \qquad \lambda_2 = \delta^t, \qquad k' = t + H(\lambda_1, \lambda_2)u,\ k' = L'L + k \\ \lambda_3 &= \gamma^{L'}, \qquad \lambda_4 = \delta^{L'} \end{aligned}$$

C veröffentlicht $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, k)$.
B verifiziert, dass gilt:

$$\gamma^k = \lambda_1 \beta^{H(\lambda_1, \lambda_2)} \lambda_3^{-L} \quad \text{und} \tag{3.9}$$

$$\delta^k = \lambda_2 \zeta^{H(\lambda_1, \lambda_2)} \lambda_4^{-L} \tag{3.10}$$

Anschließend kann jedermann mit der Kenntnis von M, der Signatur $(\delta, e, \xi_2, \xi_3)$ und mit dem nachträglich veröffentlichten Quintupel $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, k)$ verifizieren, dass die Signatur korrekt ist: Man hat hierzu $\zeta = e \oplus H(M, D(e, \xi_2, \xi_3, \alpha))$ und (3.9) zu verifizieren.

DC-Schnorr - Konvertierbare Signaturen mit designiertem Beglaubiger basierend auf der Schnorr-Identifikation In das oben beschriebene Schema setzen wir nun das Schnorr-Identifikationsprotokoll als 3-Wege-Protokoll (A_1, A_2, D) ein. Eine für endliche Abelsche Gruppen mit unbekannter Ordnung modifizierte Variante lautet wie folgt.
A hat den öffentlichen Schlüssel $\alpha = \gamma^s$, und den geheimen Schlüssel s. **A** authentifiziert sich wie folgt bei **B**:
A sendet $\xi_1 = \gamma^w\ (= A_1(w))$ mit zufällig gewähltem $w \in \{1, \ldots, L-1\}$.
B sendet ein zufälliges $e \in \{1, \ldots, L-1\}$ an **A**.
A berechnet $\xi_2', \xi_2, L' \in \mathbb{Z}, \xi_3 \in G$ mit $\xi_2' = w + es$, $\xi_2' = L'L + \xi_2$, $0 \leqslant \xi_2 < L$, $\xi_3 = \gamma^{L'}$.
A sendet an **B**: $(\xi_2, \xi_3)(= A_2(w, e, s))$
B verifiziert, dass $\xi_1 = \gamma^{\xi_2} \alpha^e \xi_3^{-L}\ (= D(e, \xi_2, \xi_3, \alpha))$.

Es ergibt sich das folgende konvertierbare Signaturverfahren mit designiertem Beglaubiger (DC-Schnorr):

Schlüsselerzeugung: **A** wählt eine endliche Abelsche Gruppe G, ein zufälliges Element $\gamma \in G$ und eine Zahl L in der Größenordnung der Gruppenordnung: $L \approx |G|, L \geqslant |G|$;
A wählt eine Zufallszahl s aus $\{1, \ldots, L-1\}$ und berechnet $\alpha = \gamma^s$.

A hat den öffentlichen Schlüssel (G, L, γ, α) und den geheimen Schlüssel s.
Der von **A** vorab bestimmte Beglaubiger **C** (designated confirmer) wählt einen zufälligen geheimen Schlüssel $u \in \{1, \ldots, L-1\}$ und den öffentlichen Schlüssel $\beta = \gamma^u$.

Signatur: **A** führt die folgenden Schritte aus:
Wähle Zufallszahlen r, w aus $\{1, \ldots, L-1\}$;
berechne $\delta = \gamma^r$;
berechne $\xi_1 = \gamma^w (= A_1(w))$;
berechne $e = \beta^r \oplus H(M, \xi_1)$;
berechne $\xi_2', \xi_2, L' \in \mathbb{Z}, \xi_3 \in G$ mit $\xi_2' = w + es$, $\xi_2' = L'L + \xi_2$, $0 \leqslant \xi_2 < L$, $\xi_3 = \gamma^{L'}$; $((\xi_2, \xi_3) = A_2(w, e, s))$;

Die Signatur des Textes M mit designiertem Beglaubiger **C** ist $Sig_{\mathbf{A},M,DC-Schnorr\ \mathbf{C}} = (\delta, e, \xi_2, \xi_3)$.

Verifikation (**A** kooperiert mit **B**):
A (der Signierer) und **B** (der Verifizierer) berechnen $\zeta = e \oplus H(M, \gamma^{\xi_2}\alpha^e\xi_3^{-L})$ $(= e \oplus H(M, D(e, \xi_2, \xi_3, \alpha)))$.
A führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \delta = \log_\beta \zeta$ (siehe Abschnitt 3.3.5.2).
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Beglaubigung (**C** kooperiert mit **B** zum Gültigkeitsnachweis der Signatur):
C (der Beglaubiger) erhält M und die Signatur $(\delta, e, \xi_2, \xi_3)$ von **A** oder **B**.
C und **B** (der Verifizierer) berechnen $\zeta = e \oplus H(M, \gamma^{\xi_2}\alpha^e\xi_3^{-L})$.
C führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \beta = \log_\delta \zeta$ (siehe Abschnitt 3.3.5.2).
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Konvertierung einer Signatur $(\delta, e, \xi_2, \xi_3)$ für den Text M durch **C**:
C berechnet $\zeta = e \oplus H(M, \gamma^{\xi_2}\alpha^e\xi_3^{-L})$.
C führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \beta = \log_\delta \zeta$ (siehe Abschnitt 3.3.5.2).
C wählt $t \in \{1, \ldots, L-1\}$ zufällig.
C berechnet $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in G$ und $k', L', k \in \mathbb{Z}$ mit $0 \leqslant k < L$ und

$$\begin{aligned} \lambda_1 &= \gamma^t, \qquad \lambda_2 = \delta^t, \qquad k' = t + H(\lambda_1, \lambda_2)u,\ k' = L'L + k \\ \lambda_3 &= \gamma^{L'}, \qquad \lambda_4 = \delta^{L'} \end{aligned}$$

C veröffentlicht $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, k)$.
B verifiziert, dass gilt:

$$\gamma^k = \lambda_1 \beta^{H(\lambda_1,\lambda_2)} \lambda_3^{-L} \quad \text{und} \tag{3.11}$$

$$\delta^k = \lambda_2 \zeta^{H(\lambda_1,\lambda_2)} \lambda_4^{-L} \tag{3.12}$$

Anschließend kann jedermann mit der Kenntnis von M, der Signatur $(\delta, e, \xi_2, \xi_3)$ und mit dem nachträglich veröffentlichten Quintupel $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, k)$ verifizieren, dass die Signatur korrekt ist: Man hat hierzu $\zeta = e \oplus H(M, \gamma^{\xi_2}\alpha^e\xi_3^{-L})$ und (3.11) zu verifizieren.

DC-GQ - Konvertierbare Signaturen mit designiertem Beglaubiger basierend auf der Guillou-Quisquater-Identifikation In das oben beschriebene Schema setzen wir nun

die Ohta-Okamoto-Variante [OO90] des Guillou-Quisquater-Identifikationsprotokolls als 3-Wege-Protokoll (A_1, A_2, D) ein. Das Schema lautet für endliche Abelsche Gruppen G mit unbekannter Ordnung wie folgt.
A hat den öffentlichen Schlüssel $\alpha = \sigma^{-s}$ und den geheimen Schlüssel σ ($s \in \{1, \ldots, L-1\}, \sigma \in G$). **A** authentifiziert sich wie folgt bei **B**:
A sendet $\xi_1 = \omega^s$ $(= A_1(\omega))$ mit zufällig gewähltem $\omega \in G$.
B sendet ein zufälliges $e \in \{1, \ldots, L-1\}$ an **A**.
A berechnet $\xi_2, \xi_3 \in G$ mit $\xi_2 = \omega\sigma^e$, $\xi_3 = 1$.
A sendet $(\xi_2, \xi_3)(= A_2(\omega, e, \sigma))$ an **B**
B verifiziert, dass $\xi_1 = \xi_2^s \alpha^e$ $(= D(e, \xi_2, \xi_3, \alpha))$.

Es ergibt sich das folgende konvertierbare Signaturverfahren mit designiertem Beglaubiger (DC-GQ):

Schlüsselerzeugung: **A** wählt eine endliche Abelsche Gruppe G, ein zufälliges Element $\sigma \in G$ und eine Zahl L in der Größenordnung der Gruppenordnung: $L \approx |G|, L \geqslant |G|$;
A wählt eine Zufallszahl $s \in \{1, \ldots, L-1\}$.

A hat den öffentlichen Schlüssel (G, L, s, α) mit $\alpha = \sigma^{-s}$ und den geheimen Schlüssel σ.

Der von **A** vorab bestimmte Beglaubiger **C** (designated confirmer) wählt einen zufälligen geheimen Schlüssel $u \in \{1, \ldots, L-1\}$ und den öffentlichen Schlüssel (β, γ) mit $\gamma \in G$ und $\beta = \gamma^u$.

Signatur: **A** führt die folgenden Schritte aus:
Wähle ein zufälliges Element $\omega \in G$ und eine Zufallszahl r aus $\{1, \ldots, L-1\}$;
berechne $\delta = \gamma^r$;
berechne $\xi_1 = \omega^s (= A_1(\omega))$;
berechne $e = \beta^r \oplus H(M \| \xi_1)$;
berechne $\xi_2 = \omega\sigma^e$, $\xi_3 = 1$.

Die Signatur des Textes M mit designiertem Beglaubiger **C** ist $Sig_{\mathbf{A},M,DC-GQ\,\mathbf{C}} = (\delta, e, \xi_2, \xi_3)$.

Verifikation (**A** kooperiert mit **B**):
A (der Signierer) und **B** (der Verifizierer) berechnen $\zeta = e \oplus H(M \| \xi_2^s \alpha^e)$ $(= e \oplus H(M \| D(e, \xi_2, \xi_3, \alpha)))$.
A führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \delta = \log_\beta \zeta$ (siehe Abschnitt 3.3.5.2).
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Beglaubigung (**C** kooperiert mit **B** zum Gültigkeitsnachweis der Signatur):
C (der Beglaubiger) erhält M und die Signatur $(\delta, e, \xi_2, \xi_3)$ von **A** oder **B**.
C und **B** (der Verifizierer) berechnen $\zeta = e \oplus H(M \| \xi_2^s \alpha^e)$.
C führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \beta = \log_\delta \zeta$ (siehe Abschnitt 3.3.5.2).
B akzeptiert die Signatur genau dann, wenn **B** den Zero-Knowledge-Beweis akzeptiert.

Konvertierung einer Signatur $(\delta, e, \xi_2, \xi_3)$ für den Text M durch **C**:
C berechnet $\zeta = e \oplus H(M \| \xi_2^s \alpha^e)$.
C führt einen Zero-Knowledge-Beweis, dass $\log_\gamma \beta = \log_\delta \zeta$ (siehe Abschnitt 3.3.5.2).

C wählt $t \in \{1, \ldots, L-1\}$ zufällig.
C berechnet $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in G$ und $k', L', k \in \mathbb{Z}$ mit $0 \leqslant k < L$ und

$$\begin{aligned} \lambda_1 &= \gamma^t, \qquad \lambda_2 = \delta^t, \qquad k' = t + H(\lambda_1 \| \lambda_2)u,\ k' = L'L + k \\ \lambda_3 &= \gamma^{L'}, \qquad \lambda_4 = \delta^{L'} \end{aligned}$$

C veröffentlicht $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, k)$.
B verifiziert, dass gilt:

$$\gamma^k = \lambda_1 \beta^{H(\lambda_1 \| \lambda_2)} \lambda_3^{-L} \quad \text{und} \tag{3.13}$$

$$\delta^k = \lambda_2 \zeta^{H(\lambda_1 \| \lambda_2)} \lambda_4^{-L} \tag{3.14}$$

Anschließend kann jedermann mit der Kenntnis von M, der Signatur $(\delta, e, \xi_2, \xi_3)$ und mit dem nachträglich veröffentlichten Quintupel $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, k)$ verifizieren, dass die Signatur korrekt ist: Man hat hierzu $\zeta = e \oplus H(M \| \xi_2^s \alpha^e)$ und (3.13) zu verifizieren.

Über die Sicherheit der vorgestellten Signaturverfahren mit designiertem Beglaubiger ist nichts bekannt. Eine entsprechende Analyse (basierend auf der Definition aus [Oka94]) bleibt Gegenstand künftiger Forschung. Eine analoge Situation treffen wir für die in [Oka94] vorgestellten Verfahren an.

3.3.5.3 Blinde Signaturverfahren

Anders als bei normalen Signaturverfahren gibt es in *Blinden Signaturverfahren (blind signature schemes)* drei Parteien, einen Sender **B**, einen Signierer **A** und einen Verifizierer **C**. In Blinden Signaturverfahren cachiert **B** den von **A** zu signierenden Text M *(Blinding)* und legt **A** die Cachierung $g(M)$ des Textes vor. **A** signiert die Cachierung $g(M)$, ohne hieraus M bestimmen zu können. Man sagt, „**A** signiere blind". **B** kann aus **A**s Signatur für die Cachierung $g(M)$ des Textes eine gültige Signatur **A**s für den ursprünglich gewählten Text M berechnen *(Unblinding)*. Diese wird von einem beliebigen Verifizierer **C** akzeptiert. Der Signierer **A** weiß am Ende weder den tatsächlich signierten Text M (er kennt nur die Cachierung $g(M)$) noch kann er feststellen, auf welche Cachierung und auf welchen Sender eine gültige Signatur zurück geht.

Blinde Signaturverfahren finden Anwendung in digitalen Bezahlverfahren (digital cash) und bei elektronischen Wahlen. Bei Digital Cash-Protokollen repräsentieren Texte M digitale Münzen. Der Bankkunde **B** möchte nicht, dass die Bank (Signierer **A**) später nachvollziehen kann, dass eine für eine digitale Münze gültige Bank-Signatur auf den Kunden **B** zurück geht. In elektronischen Wahlverfahren generiert der Wähler **B** zunächst alle theoretisch möglichen Stimmzettel, cachiert diese (Blinding) und sendet diese an eine vertrauenswürdige Instanz (Wahlleiter) **A**, welche alle cachierten Stimmzettel signiert und an den Wähler **B** zurück sendet. **B** legt sich nun auf einen der von **A** signierten Stimmzettel seiner Wahl fest (Unblinding) und sendet diesen an **A** zurück. **A** erkennt, dass es sich um einen gültigen elektronischen Stimmzettel handelt (Verifikation der Signatur), kann aber nicht feststellen, von welchem Wähler dieser Stimmzettel stammt. Detaillierte Beschreibungen der Anwendungsmöglichkeiten von Blinden Signaturverfahren finden sich in [Cha83] und [Sch94].

Wir stellen ein Blindes Signaturverfahren vor, welches in beliebigen endlichen Abelschen Gruppen implementiert werden kann, in denen man effizient die Gruppenoperation ausführen, Gruppenelemente invertieren und die Gleichheit zweier Gruppenelemente entscheiden kann. Die Kenntnis

der Gruppenordnung ist nicht notwendig; allerdings sollte eine (obere) Abschätzung der Gruppenordnung vorliegen. Das Blinde Signaturverfahren ist sicher, wenn das Wurzelproblem in der entsprechenden Gruppe schwer ist. Somit eignen sich die von uns in Abschnitt 4.3 vorgestellten Klassengruppen algebraischer Zahlkörper bestens zur Implementierung.

Blinde Signaturverfahren wurden erstmals von Chaum vorgeschlagen und realisiert [Cha83, Cha85]. In [CP93] wird ein Blindes Signaturverfahren vorgeschlagen, in welchem ein Zero-Knowledge-Beweis über die Kenntnis eines diskreten Logarithmus geführt wird. Die Arbeit [CPS95] zeigt, wie die DL-basierten Signaturverfahren ElGamal [ElG85] und Nyberg-Rueppel [NR93] zu Blinden Signaturverfahren erweitert werden können. Das Konstruktionsverfahren aus [CP93] lässt sich nur in Gruppen mit bekannter primer Ordnung anwenden – Klassengruppen oder Untergruppen von Klassengruppen mit primer Ordnung können wir allerdings nicht erzeugen. Auch die Verfahren aus [CPS95] benötigen die Kenntnis der primen Ordnung einer Untergruppe. Zudem arbeitet das genannte Verfahren in einer Untergruppe der primen Restklassengruppe modulo einer Primzahl. Deswegen können Gruppenelemente als ganze Zahlen interpretiert und somit im Protokoll als Exponenten verwendet werden. Basierend auf Ideen aus [CPS95] zeigen wir im Folgenden, wie das ElGamal-Signaturverfahren zu einem Blinden Signaturverfahren erweitert werden kann, welches die Kenntnis der (Unter-)Gruppenordnung und die Konstruktion einer Untergruppe mit primer Ordnung nicht erfordert. Herzu folgen wir der beim Design des RDSA-Protokolls (siehe Abschnitt 3.3.4.1) angewandten Grundidee: Wir ersetzen die prime Untergruppenordnung im Protokoll [CPS95] durch eine beliebige, aber fixe große Primzahl q und reduzieren die im Protokoll vorkommenden Exponenten modulo q. Zudem führen wir eine Hashfunktion h ein, die Gruppenelemente auf natürliche Zahlen abbildet.

Mit diesen massiven Protokolländerungen ist a priori keineswegs klar, ob die Sicherheitsaussagen aus [CPS95] auch für unser neues Verfahren gelten. Daher diskutieren wir weiter unten die Sicherheit unseres Verfahrens explizit. Im Vergleich zu [CPS95] erhalten wir dabei tatsächlich analoge Sicherheitsaussagen für unser neues Verfahren.

Die Sicherheit unseres Verfahrens basiert auf dem Wurzelproblem, einem Spezialfall des diskreten Logarithmusproblems (siehe Abschnitt 3.1). Daher nennen wir unser Verfahren Blind-RDSA.

Wir definieren zunächst, wann ein Signaturverfahren *blind* heißt. Dann präsentieren wir das zugrunde liegende Signaturverfahren, eine RDSA-Variante. Anschließend beschreiben wir das darauf aufbauende Blinde Signaturverfahren Blind-RDSA. Wir beweisen die Abgeschlossenheit von Blind-RDSA und zeigen, dass Blind-RDSA blind ist.

Wir definieren ein Blindes Signaturverfahren gemäß [CPS95]:
Ein Signaturverfahren heißt *blind*, wenn des Signierers **A** komplette Sicht einer Ausführung des Protokolls (d.h. Sicht auf die Wahl von zufälligen Objekten und auf alle ausgetauschten Informationen) und die Text-Signatur-Paare $(M, Sig_{A,M})$ statistisch unabhängig voneinander sind.

RDSA-Variante – Das Verfahren

Wir stellen die RDSA-Variante vor, die sich aus [MOV97, S. 457, Signaturgleichung 3, Tab. 11.5] ergibt. Sei $h : G \to \{1, \ldots, q-1\}$ eine kryptographische Hashfunktion.

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:

Wähle eine endliche Abelsche Gruppe G und eine hinreichend große zufällige Primzahl q mit $2^{t-1} < q \leqslant 2^t$ für einen Sicherheitsparameter $t \in \mathbb{Z}_{>0}$ (z.B. $t = 160$);

wähle eine gute Näherung $L \approx |G|$ mit $|G| \leqslant L$, $L \in \mathbb{Z}_{>0}$;
wähle ein zufälliges Element $\gamma \in G$;
wähle eine Zufallszahl a aus $\{2, \ldots, L-1\}$ und berechne $\alpha = \gamma^a$.

As öffentlicher Schlüssel ist (G, γ, α, q), **A**s privater Schlüssel ist a.

Signatur für Text $M \in \{1, \ldots, q-1\}$: A führt die folgenden Schritte aus:
Wähle eine Zufallszahl k aus $\{1, \ldots, L-1\}$;
berechne die Gruppenelemente $\varrho = \gamma^k$ und $\mu = 1$;
berechne $h(\varrho)$;
berechne ganze Zahlen s und ℓ, so dass $kM + h(\varrho)a = \ell q + s$ und $0 \leqslant s < q$;
berechne $\lambda = \gamma^\ell$.

Die Signatur des Textes M ist $Sig^{RDSA}_{\mathbf{A},M} = (s, \varrho, \lambda, \mu)$.

Verifikation: Ein Verifizierer akzeptiert die Signatur $Sig^{RDSA}_{\mathbf{A},M}$ genau dann, wenn $0 \leqslant s < q$ und $\gamma^s \lambda^q = \alpha^{h(\varrho)} \varrho^M \cdot \mu^M$ ist.

Das Signaturverfahren ist abgeschlossen (complete). Wenn nämlich ein Signierer **A** und ein Verifizierer dem Protokoll folgen, wird die Signatur $Sig^{RDSA}_{\mathbf{A},M}$ durch **B** akzeptiert:

$$\gamma^s \lambda^q = \gamma^{ah(\varrho)+kM-\ell q} \cdot \gamma^{\ell q} = \alpha^{h(\varrho)} \varrho^M \mu^M \quad . \tag{3.15}$$

Blind-RDSA – Blindes Signaturverfahren basierend auf RDSA

Wir beschreiben das auf obiger RDSA-Variante aufbauende Blinde Signaturverfahren.

Signaturerstellung, erster Teil: A führt die folgenden Schritte aus:
Wähle eine Zufallszahl $\tilde{k}$ aus $\{1, \ldots, L-1\}$;
berechne die Gruppenelemente $\tilde{\varrho} = \gamma^{\tilde{k}}$ und $\tilde{\mu} = 1$;
sende $(\tilde{\varrho}, \tilde{\mu})$ an **B**.

Blinding des Textes $M \in \{1, \ldots, q-1\}$ prim: B führt die folgenden Schritte aus:
Wähle Zufallszahlen b_1, b_2 aus $\{1, \ldots, L-1\}$;
berechne $\varrho = \tilde{\varrho}^{b_1} \gamma^{b_2}$;
berechne ganze Zahlen $h(\varrho)', q_\varrho$ mit $h(\varrho)h(\varrho)' = 1 + q_\varrho q$;
berechne ganze Zahlen $\tilde{M}, q_2$ mit $b_1 M h(\tilde{\varrho})h(\varrho)' = q_2 q + \tilde{M}$ und $0 \leqslant \tilde{M} < q$;
sende $\tilde{M}$ an **A**.

Signaturerstellung, zweiter Teil: A führt die folgenden Schritte aus:
Berechne ganze Zahlen $\tilde{s}$ und $\tilde{\ell}$, so dass $\tilde{k}\tilde{M} + h(\tilde{\varrho})a = \tilde{\ell} q + \tilde{s}$ und $0 \leqslant \tilde{s} < q$;
berechne $\tilde{\lambda} = \gamma^{\tilde{\ell}}$.

A sendet die Signatur des Textes $\tilde{M}$ an Bob: $Sig^{Blind}_{\mathbf{A},\tilde{M}} = (\tilde{s}, \tilde{\varrho}, \tilde{\lambda}, \tilde{\mu})$.

Unblinding des Textes: B führt die folgenden Schritte aus:
Berechne ganze Zahlen $h(\tilde{\varrho})', q_{\tilde{\varrho}}$ mit $h(\tilde{\varrho})h(\tilde{\varrho})' = 1 + q_{\tilde{\varrho}} q$;
berechne ganze Zahlen s und ℓ, so dass $\tilde{s} h(\varrho) h(\tilde{\varrho})' + b_2 M = \ell q + s$ und $0 \leqslant s < q$;

berechne $z = (1 + q_{\tilde{\varrho}}q)(1 + q_\varrho q) - 1$;
berechne $\lambda = \alpha^{-h(\varrho)q_{\tilde{\varrho}}}\tilde{\varrho}^{q_2 h(\varrho)h(\tilde{\varrho})'}\gamma^{\ell}\tilde{\lambda}^{h(\tilde{\varrho})'h(\varrho)}$;
berechne $\mu = \tilde{\varrho}^{b_1 z}$.

Die „von **A** ausgestellte" blinde Signatur des Textes M ist $Sig^{Blind}_{\mathbf{A},M} = (s, \varrho, \lambda, \mu)$.

Das Signaturverfahren Blind-RDSA ist abgeschlossen (complete):

Proposition 3.3.17
*Wenn **A**, **B** und ein Verifizierer **C** dem vorstehenden Blind-RDSA-Protokoll folgen, dann ist die Verifikation einer blinden Signatur erfolgreich.*

Beweis: Mit den Bezeichnungen von oben gilt

$$\begin{aligned}\tilde{k}h(\varrho)h(\tilde{\varrho})' \cdot (\tilde{M} + q_2 q) &= \tilde{k}b_1 M h(\varrho)h(\varrho)'h(\tilde{\varrho})h(\tilde{\varrho})' \\ &= \tilde{k}b_1 M(1 + q_\varrho q)(1 + q_{\tilde{\varrho}}q) \\ &= \tilde{k}b_1 M(z + 1) \\ &= \tilde{k}b_1 M + \tilde{k}b_1 z M \ .\end{aligned} \tag{3.16}$$

Außerdem gilt

$$\begin{aligned}\gamma^s\lambda^q &= \gamma^{\tilde{s}h(\varrho)h(\tilde{\varrho})' + b_2 M - \ell q} \cdot \gamma^{-ah(\varrho)q_{\tilde{\varrho}}q + \tilde{k}q_2 q h(\varrho)h(\tilde{\varrho})' + \ell q + \tilde{\ell}q h(\tilde{\varrho})'h(\varrho)} \\ &= \gamma^{\tilde{k}\tilde{M}h(\varrho)h(\tilde{\varrho})' + ah(\varrho)(1+q_{\tilde{\varrho}}q) - \tilde{\ell}q h(\varrho)h(\tilde{\varrho})' + b_2 M - \ell q - ah(\varrho)q_{\tilde{\varrho}}q + \tilde{k}q_2 q h(\varrho)h(\tilde{\varrho})' + \ell q + \tilde{\ell}q h(\tilde{\varrho})'h(\varrho)} \\ &\overset{(3.16)}{=} \gamma^{ah(\varrho) + \tilde{k}b_1 M + b_2 M + \tilde{k}b_1 z M} \\ &= \alpha^{h(\varrho)}\varrho^M \mu^M \ .\end{aligned}$$

□

Es sei bemerkt, dass die Forderung, der vom Sender cachierte Text M solle prim sein, durch den Sender **B** wie folgt gewährleistet werden kann: Wenn **B** eine gültige Signatur **A**s für eine Zeichenkette erhalten möchte, deren dezimale Repräsentation M nicht prim ist, so hängt **B** an die Zeichenkette so viele Sonderzeichen oder Leerzeichen an, bis die dezimale Repräsentation M prim ist.

Wir beweisen nun die Blindheit des Signaturverfahrens Blind-RDSA:

Satz 3.3.18
Das Signaturverfahren Blind-RDSA ist blind.

Beweis: Wir zeigen, dass es für eine gegebene komplette Sicht von **A** auf die Ausführung des Protokolls und ein gegebenes gültiges Text-Signatur-Paar $(M, Sig^{Blind}_{\mathbf{A},M})$ genau ein Paar Blinding-Faktoren $b_1, b_2 \in \{1, \ldots, q\}$ und konstant viele Paare $b_1, b_2 \in \{1, \ldots, |G|\}$ gibt. d.h. die Anzahl der Paare Blinding-Faktoren aus $\{1, \ldots, |G|\}$ ist unabhängig von der Wahl des Text-Signatur-Paars. Da der Sender **B** die Blinding-Faktoren b_1, b_2 zufällig aus $\{1, \ldots, L - 1 \approx |G|\}$ auswählt, folgt die statistische Unabhängigkeit und damit die Blindheit des Signaturverfahrens.

Während einer Ausführung des Protokolls sei für den Text M die gültige Signatur $Sig^{Blind}_{\mathbf{A},M} = (s, \rho, \lambda, \mu)$ entstanden. **A**s Sicht auf diese Ausführung des Protokolls bestehe aus

$$\tilde{k}, \tilde{\rho} = \gamma^{\tilde{k}}, \tilde{\mu} = 1, \tilde{M}, \tilde{s}, \tilde{\ell} \text{ mit } \tilde{k}\tilde{M} + h(\tilde{\varrho})a = \tilde{\ell}q + \tilde{s}, \tilde{\lambda} = \gamma^{\tilde{\ell}}.$$

Dann müssen die folgenden Bedingungen für $b_1, b_2 \in \{1, \ldots, |G|\}$ gelten:

$$\begin{aligned} b_1 M h(\tilde{\varrho}) h(\varrho)' &= q_2 q + \tilde{M} \text{ und } 0 \leqslant \tilde{M} < q && (3.17)\\ \tilde{s} h(\varrho) h(\tilde{\varrho})' + b_2 M &= \ell q + s \text{ und } 0 \leqslant s < q && (3.18)\\ \varrho\mu &= \tilde{\varrho}^{b_1} \gamma^{b_2} \tilde{\varrho}^{b_1 \cdot [(1+q_{\tilde{\varrho}} q)(1+q_\varrho q)-1]} && (3.19)\\ h(\varrho) h(\varrho)' &= 1 + q_\varrho q && (3.20)\\ h(\tilde{\varrho}) h(\tilde{\varrho})' &= 1 + q_{\tilde{\varrho}} q && (3.21) \end{aligned}$$

Durch die Bedingungen (3.20),(3.21) sind $h(\varrho)', q_\varrho, h(\tilde{\varrho})', q_{\tilde{\varrho}}$ eindeutig bestimmt. Daher sind durch die Bedingungen (3.17),(3.18) die Blinding-Faktoren b_1, b_2 modulo q eindeutig bestimmt: Da $M, h(\tilde{\varrho}), h(\varrho)'$ jeweils teilerfremd zu q sind, folgt aus (3.17)

$$\begin{aligned} b_1 &\equiv M^{-1} \tilde{M} h(\tilde{\varrho})' h(\varrho) \mod q,\\ b_1 M \overbrace{(1 + q_{\tilde{\varrho}} q)(1 + q_\varrho q)}^{=z+1} &= \tilde{M} h(\tilde{\varrho})' h(\varrho) + h(\tilde{\varrho})' h(\varrho) q_2 q\\ \text{und } b_1 M &= \tilde{M} h(\tilde{\varrho})' h(\varrho) + h(\tilde{\varrho})' h(\varrho) q_2 q - b_1 M z \ . \end{aligned}$$

Entsprechend folgt aus (3.17)

$$\begin{aligned} b_2 &\equiv M^{-1}(s - \tilde{s} h(\varrho) h(\tilde{\varrho})') \mod q\\ \text{und } b_2 M &= s - \tilde{s} h(\varrho) h(\tilde{\varrho})' + \ell q \ . \end{aligned}$$

Wir haben noch zu verifizieren, dass die durch (3.20),(3.21),(3.17),(3.18) modulo q eindeutig bestimmten Werte b_1, b_2 auch die Bedingung (3.19) erfüllen. Das folgende Hilfsresultat ergibt sich durch (3.21) und wegen $\tilde{s} = -\ell q + \tilde{k} \tilde{M} h(\tilde{\varrho}) a$:

$$\begin{aligned} \tilde{k} b_1 M + b_2 M &= (\tilde{k} \tilde{M} h(\tilde{\varrho})' h(\varrho) + \tilde{k} h(\tilde{\varrho})' h(\varrho) q_2 q - \tilde{k} b_1 M z) + (s - \tilde{s} h(\varrho) h(\tilde{\varrho})' + \ell q)\\ &= \tilde{k} \tilde{M} h(\tilde{\varrho})' h(\varrho) + \tilde{k} h(\tilde{\varrho})' h(\varrho) q_2 q - \tilde{k} b_1 M z + s - \tilde{k} \tilde{M} h(\varrho) h(\tilde{\varrho})'\\ &\quad - a h(\varrho)(1 + q_{\tilde{\varrho}} q) + h(\varrho) h(\tilde{\varrho})' \tilde{\ell} q + \ell q\\ &= s - a h(\varrho) - a h(\varrho) q_{\tilde{\varrho}} q + \tilde{\ell} h(\varrho) h(\tilde{\varrho})' q + \ell q + \tilde{k} q_2 h(\varrho) h(\tilde{\varrho})' q - \tilde{k} b_1 z M \end{aligned}$$

Daher gilt für die durch (3.20),(3.21),(3.17),(3.18) modulo q eindeutig bestimmten Werte b_1, b_2:

$$\begin{aligned} (\tilde{\varrho}^{b_1} \gamma^{b_2})^M &= \gamma^{\tilde{k} b_1 M + b_2 M}\\ &= \gamma^s \cdot \alpha^{-h(\varrho)} \cdot (\alpha^{-h(\varrho) q_{\tilde{\varrho}}})^q \cdot (\tilde{\lambda}^{h(\varrho) h(\tilde{\varrho})'})^q \cdot (\gamma^\ell)^q \cdot (\tilde{\rho}^{q_2 h(\varrho) h(\tilde{\varrho})'})^q \cdot (\gamma^{\tilde{k} b_1 z})^{-M} \end{aligned}$$

Diese Gleichung ist äquivalent zu

$$\begin{aligned} [(\tilde{\varrho}^{b_1} \gamma^{b_2}) \cdot \gamma^{\tilde{k} b_1 z}]^M &= \gamma^s \cdot \alpha^{-h(\varrho)} \cdot \lambda^q\\ &\overset{(\text{Blind-RDSA abgeschlossen})}{=} (\varrho \cdot \mu)^M \end{aligned}$$

Da M nach Voraussetzung prim ist, können wir o.B.d.A. annehmen, dass $ggT(M, |G|) = 1$. Es folgt Gleichung (3.19). □

3.3.5.4 Identifikationsverfahren: R-Fiat-Shamir, R-Schnorr und weitere

R-Fiat-Shamir-Identifikationsprotokoll

Wir stellen nun unser R-Fiat-Shamir-Identifikationsprotokoll (R-FS) vor. Unser Protokoll basiert auf der Variante [DGB88] des Original-Fiat-Shamir-Protokolls [FS87], welche bei gleicher Sicherheit und gleichem Aufwand für den Beweiser eine effizientere Verifikation zulässt als das Original-Protokoll. Die Sicherheit das Original-Fiat-Shamir-Protokolls [FS87] und dessen Varianten wie z.B. [DGB88] basiert auf dem algorithmischen Problem, Quadratwurzeln in primen Restklassengruppen zu berechnen. Hingegen stellen wir mit R-Fiat-Shamir ein verallgemeinertes Protokoll dar, das in allen endlichen Abelschen Gruppen implementiert werden kann, in welchen die „Basisoperationen" (siehe Beginn des Kapitels 3) effizient durchführbar sind. Die Sicherheit unseres Protokolls basiert auf dem Wurzelproblem. Das Problem, Quadratwurzeln zu berechnen kann als spezielles Wurzelproblem aufgefasst werden. Das Original-Protokoll [FS87] bzw. dessen Variante [DGB88] ist daher als Spezialfall in unserem verallgemeinerten Protokoll enthalten.

Das Protokoll (Basisversion)

Der Exponent $k \in \mathbb{Z}_{\geqslant 2}$ sowie die Rundenanzahl $t \in \mathbb{Z}_{>0}$ sind für alle Teilnehmer fest vorgegeben (siehe unten, Parameterempfehlung).

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle eine endliche Abelsche Gruppe G;
wähle ein zufälliges Element $\sigma \in G$;
berechne $\alpha = \sigma^k$.

As öffentlicher Schlüssel ist (G, α), **A**s privater Schlüssel ist σ.

Identifikation: Folgende Schritte werden t Mal wiederholt:

A wählt ein zufälliges Gruppenelement $\varrho \in G$ *(Commitment)*;
A berechnet das Gruppenelement $\xi = \varrho^k$ und sendet ξ an **B** *(Zeuge, Witness)*;
B wählt ein Zufallsbit $e \in \{0, 1\}$ und sendet e an **A** *(Challenge)*;
A berechnet das Gruppenelement $\eta = \varrho\sigma^e$ und sendet η an **B** *(Response)*.
B verifiziert, dass $\eta^k = \alpha^e\xi$.

B akzeptiert **A**s Identität genau dann, wenn die Verifikation in allen t Runden positiv ausging.

Das R-Fiat-Shamir-Identifikationsverfahren (Basisversion) ist abgeschlossen (complete), wie die folgende Proposition zeigt.

Proposition 3.3.19
Wenn ***A*** *und* ***B*** *dem R-Fiat-Shamir-Identifikationsprotokoll (Basisversion) folgen, dann ist die Verifikation immer erfolgreich.*

Beweis: Mit den Bezeichnungen von oben gilt in jeder Runde $\eta^k = \varrho^k(\sigma^k)^e = \xi\alpha^e$. □

Sicherheit (Basisversion)

Die Sicherheit unseres R-Fiat-Shamir-Identifikationsverfahrens (Basisversion) basiert auf dem Problem, k-te Wurzeln zu berechnen (Wurzelproblem, Root Problem). R-FS steht für „Root Based Fiat Shamir Identification Protocol“. Wer effizient k-te Wurzeln berechnen kann, bestimmt aus dem öffentlichen Schlüssel eines beliebigen Teilnehemers direkt dessen privaten Schlüssel durch Berechnung der k-ten Wurzel.

Wir diskutieren weitere Aspekte der Sicherheit des R-Fiat-Shamir-Identifikationsverfahrens (Basisversion).

Betrugswahrscheinlichkeit. Errät ein Angreifer **E** die Challenge e in allen t Runden korrekt, so kann sich **E** erfolgreich als beliebiger Teilnehmer **A** ausgeben. **E** wählt hierzu ein beliebiges $\eta \in G$, rät e, berechnet $\xi = \alpha^{-e}\eta^k$ und sendet ξ als Zeugen und η als Response. Bei Anwendung dieses Angriffes hat **E** eine Erfolgschance von 2^{-t}. Ein besserer Angriff ist nicht bekannt, wie sich als Spezialfall aus dem unten bewiesenen Lemma 3.3.21 ergibt. Je größer also die Rundenanzahl t, umso kleiner die Betrugschance für einen Angreifer.

Soundness. R-FS ist sound. Zum Beweis nehmen wir an, ein Angreifer **E** könne sich mit Erfolgswahrscheinlichkeit 1 als **A** ausgeben. Dann kann **E** in jeder Runde auf beide mögliche Challenges ($e = 0$ und $e = 1$) eine korrekte Response geben: Für einen von **E** gewählten Zeugen ξ sei η_1 die Antwort auf $e = 0$ und η_2 die Antwort auf $e = 1$. Folglich ist $\eta_1^k = \xi$ und $\eta_2^k = \alpha\xi$ (d.h. $\eta_2^k\alpha^{-1} = \xi$). Es folgt $(\eta_1^{-1}\eta_2)^k = \alpha$, d.h. **E** kennt eine k-te Wurzel von **A**s öffentlichem Schlüssel α.

Zero-Knowledge-Eigenschaft. R-FS ist Zero-Knowledge; denn für den Beweiser **A** und Verifizierer **B** existiert der folgende Polynomzeit-Simulator: Der Simulator wählt ein zufälliges $\tilde{\eta} \in G$, wählt ein Zufallsbit $\tilde{e} \in \{0, 1\}$, berechnet $\tilde{\xi} = \alpha^{-\tilde{e}}\tilde{\eta}^k$ und gibt $(\tilde{\xi}, \tilde{e}, \tilde{\eta})$ aus. Die Verteilung des Simulators ist polynomiell ununterscheidbar von der Wahrscheinlichkeitsverteilung für eine Ausgabe (ξ, e, η) als Resultat einer Identifikationsrunde zwischen **A** und **B**. Auch das Verhalten eines betrügerischen Verifizierers, der dem Protokoll nicht folgt, kann simuliert werden.

Wir beschreiben nun die parallele Version von R-FS.

Das Protokoll (Parallele Version)

Der Exponent $k \in \mathbb{Z}_{\geqslant 2}$ und die Parameter $m, t \in \mathbb{Z}_{>0}$ sind für alle Teilnehmer fest vorgegeben (siehe unten, Parameterempfehlung).

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:
Wähle eine endliche Abelsche Gruppe G;
wähle für $j \in \{1, \ldots, m\}$ zufällige Elemente $\sigma_j \in G$ und berechne $\alpha_j = \sigma_j^k$.

As öffentlicher Schlüssel ist $(G, \alpha_1, \ldots, \alpha_m)$, **A**s privater Schlüssel ist $(\sigma_1, \ldots, \sigma_m)$.

Identifikation: Folgende Schritte werden t Mal wiederholt:

A wählt ein zufälliges Gruppenelement $\varrho \in G$ *(Commitment)*;
A berechnet das Gruppenelement $\xi = \varrho^k$ und sendet ξ an **B** *(Zeuge, Witness)*;

B wählt einen zufälligen Bitvektor $(e_1, \ldots, e_m) \in \{0,1\}^m$ und sendet $(e_1, \ldots, e_m)$ an **A** *(Challenge)*;
A berechnet das Gruppenelement $\eta = \varrho \cdot \prod_{j=1}^{m} \sigma_j^{e_j}$ und sendet η an **B** *(Response)*.
B verifiziert, dass $\eta^k = \xi \cdot \prod_{j=1}^{m} \alpha_j^{e_j}$.

B akzeptiert **A**s Identität genau dann, wenn die Verifikation in allen t Runden positiv ausging.

Die parallele Version des R-Fiat-Shamir-Identifikationsverfahrens ist abgeschlossen (complete), was sich in Analogie zu 3.3.19 einfach zeigen lässt:

Proposition 3.3.20
*Wenn **A** und **B** dem R-Fiat-Shamir-Identifikationsprotokoll (Parallele Version) folgen, dann ist die Verifikation immer erfolgreich.*

Sicherheit (Parallele Version)

Wir diskutieren die Sicherheit der parallelen Version von R-FS.

Sicherheitsannahme. Die Sicherheit der parallelen Version des R-FS-Identifikationsverfahrens basiert auf dem Problem, k-te Wurzeln zu berechnen (Wurzelproblem).

Betrugswahrscheinlichkeit. Erfolgreiches Raten der Challenge $(e_1, \ldots, e_m)$ in allen t Runden ermöglicht einem Angreifer, sich erfolgreich als beliebiger Teilnehmer **A** auszugeben. Die Erfolgschance für einen solchen Angriff beträgt 2^{-mt}. Ein besserer Angriff ist nicht bekannt (siehe Lemma 3.3.21).

Soundness. Die parallele Version von R-FS ist sound. Dies folgt aus dem folgenden Lemma:

Lemma 3.3.21
*Angenommen, ein Betrüger **E** kennt nicht die σ_j aus dem privaten Schlüssel von **A** und kann nicht probabilistisch in Polynomzeit die k-te Wurzel irgendeines Produktes der Form $\prod_{j=1}^{m} \alpha_j^{c_j}$ berechnen ($c_j \in \{-1,0,1\}$, mindestens ein $c_j \neq 0$). Dann akzeptiert ein legitimer Verifizierer **B** den Versuch **E**s, sich als **A** auszugeben, höchstens mit Wahrscheinlichkeit 2^{-mt}.*

Beweis: **E** kann sich folgendermaßen erfolgreich als **A** ausgeben: Er rät die t zufälligen Bitvektoren $(e_1, \ldots, e_m)$ korrekt, wählt für jede der t Runden jeweils ein beliebiges $\eta \in G$, berechnet $\xi = \eta^k \prod_{j=1}^{m} \alpha_j^{-e_j}$, sendet ξ als Zeugen und η als Response. Die Erfolgswahrscheinlichkeit für **E** ist dabei 2^{-m} pro Runde und 2^{-mt} für das gesamte Identifikationsprotokoll. Um diese Wahrscheinlichkeit zu erhöhen, muss **E** wenigstens ein Element ξ so wählen, dass er für wenigstens ein Paar ungleicher Bitvektoren $(e'_1, \ldots, e'_m)$, $(e''_1, \ldots, e''_m)$ probabilistisch in Polynomzeit k-te Wurzeln η' von $\xi \cdot \prod_{j=1}^{m} \alpha_j^{e'_j}$ und η'' von $\xi \cdot \prod_{j=1}^{m} \alpha_j^{e''_j}$ berechnen kann. Dann gilt $(\eta'^{-1}\eta'')^k = \prod_{j=1}^{m} \alpha_j^{e''_j - e'_j}$, d.h. **E** kennt die k-te Wurzel eines Produktes der Form $\prod_{j=1}^{m} \alpha_j^{c_j}$, wobei $c_j \in \{-1,0,1\}$ und mindestens ein $c_j \neq 0$ ist. Dies widerspricht der Voraussetzung, denn **E** kann die vom Verifizierer **B** vorgegebenen zufälligen Bitvektoren $(e'_1, \ldots, e'_m)$, $(e''_1, \ldots, e''_m)$ selbst wählen und dann effizient ein Gruppenelement berechnen, von dem wir annahmen, er könne es nicht. □

Zero-Knowledge-Eigenschaft. Bei beliebiger Wahl des Parameters m und der Rundenanzahl t ist die parallele Version von R-FS im Allgemeinen nicht Zero-Knowledge. Dennoch bleibt die Sicherheit des Verfahrens auf Grund dessen Soundness-Eigenschaft gewahrt, sofern in der zugrunde liegenden Gruppe nicht effizient k-te Wurzeln berechnet werden können. Eine analoge Situation liegt bei den Original-Protokollen [FS87] und [FFS88] vor. Der Verlust der Zero-Knowledge-Eigenschaft bei der Parallelisierung von Identifikationsprotokollen ist aus der Fachliteratur bekannt als der Fall, in dem die Intuition fehlt (siehe [FFS88]).

Effizienz (Parallele Version)

Wir diskutieren die Effizienz der parallelen Version von R-FS.

Berechnungskomplexität Der Beweiser **A** hat t Gruppenelemente zufällig zu wählen und t viele k-te Potenzen zu berechnen. Hierbei ist eine Vorberechnung komplett möglich. Zudem hat **A** im Durchschnitt $\frac{mt}{2}$ (im schlechtesten Fall mt) viele Gruppenoperationen auszuführen. (Hier ist keine Vorberechnung möglich.) Der Verifizierer **B** hat ebenfalls t viele k-te Potenzen zu berechnen (keine Vorberechnung möglich) und im Durchschnitt $\frac{mt}{2}$ (im schlechtesten Fall mt) viele Gruppenoperationen auszuführen (**B** kann mit diesen Berechnungen beginnen, sobald er sich auf eine Challenge festgelegt hat.). Zudem hat der Verifizierer t viele Gleichheitsentscheide durchzuführen. Die Berechnungskomplexität kann folglich bei konstantem Sicherheitsniveau minimiert werden, indem bei konstantem mt als Rundenzahl $t = 1$ gewählt wird. Hierdurch wird auch die Kommunikationslast minimiert, allerdings erreicht der Speicherplatzbedarf für die Geheimnisse der Teilnehmer gleichzeitig ein Maximum, und die Zero-Knowledge-Eigenschaft geht verloren.

Speicherplatzbedarf für Geheimnisse Die Teilnehmer benötigen zur Speicherung ihrer Geheimnisse (private Schlüssel) Speicherplatz für m Gruppenelemente. Der Speicherplatzbedarf ist bei konstantem Sicherheitsniveau also minimal (bzw. maximal), wenn mt konstant gehalten wird und $m = 1$ (bzw. $t = 1$) verwendet wird.

Kommunikationslast In der parallelen Version von R-FS werden $2t$ Gruppenelemente und mt viele Bits übertragen. Die Kommunikationslast ist bei konstantem Sicherheitsniveau also minimal (bzw. maximal), wenn mt konstant gehalten wird und $t = 1$ (bzw. $m = 1$) verwendet wird.
Die Kommunikationslast kann zusätzlich reduziert werden, indem der Beweiser **A** als Zeugen in jeder Runde statt des Gruppenelementes ξ einen Hashwert von $h(\xi)$ schickt. Die Hashfunktion h wird in diesem Fall vorher für alle Teilnehmer festgelegt und veröffentlicht. Zur Verifikation berechnet der Verifizierer in jeder Runde $\xi' = \eta^k \cdot \prod_{j=1}^{m} \alpha_j^{-e_j}$ und prüft, ob $h(\xi) = h(\xi')$ gilt.

Empfehlung zur Parameterwahl (Parallele Version)

R-FS ist ein interaktives Identifikationsverfahren, bei welchem dem Beweiser zufällige Challenges vorgelegt werden. Für geeignete Gruppen G ist kein besserer Angriff bekannt, als die Challenges in jeder Runde zu raten. In der analogen Situation im Originalverfahren [FS87] erläutern die Autoren, ein Sicherheitsniveau von 2^{-20} reiche in der Praxis für fast alle Anwendungen aus. Wir folgen dieser Parameterempfehlung für unser Verfahren; denn Identifikationssysteme sind in der

Regel so konstruiert, dass eine kleine Anzahl an Fehlversuchen unter dem Namen eines Teilnehmers zum (temporären) Sperren des Teilnehmers führt. In Analogie zu [FS87] empfehlen wir die Wahl von $t = 1$ und $m = 20$. Diese Parameterwahl minimiert für das Sicherheitsniveau 2^{-20} sowohl die Berechnungskomplexität als auch die Kommunikationslast.

Zudem empfehlen wir die Verwendung von $k \in \{2, 3\}$, so dass die Sicherheit des R-FS-Identifikationsverfahrens auf der Schwierigkeit basiert, Quadratwurzeln bzw. Kubikwurzeln zu berechnen.

(Es sei bemerkt, dass die Berechnung von Quadratwurzeln im Spezialfall imaginär-quadratischer Zahlkörper – und nur da – polynomiell äquivalent zum Faktorisieren ganzer Zahlen[7] ist. Ein ausführlicher Beweis findet sich in [Mey97].)

R-Schnorr-Identifikationsprotokoll

Wir stellen nun unser R-Schnorr-Identifikationsprotokoll vor. Es handelt sich hier um eine Variante des Schnorr-Identifikationsprotokolls [Sch90]. Im Original-Schnorr-Identifikationsprotokoll wird in einer zyklischen Untergruppe der primen Restklassengruppe gerechnet, wobei die Untergruppe prime Ordnung hat. Zudem werden die Exponenten modulo der primen Ordnung der Untergruppe reduziert, welche allen Protokollteilnehmern bekannt ist. Wir modifizieren das Protokoll so, dass es in beliebigen endlichen Abelschen Gruppen anwendbar ist, insbesondere, wenn die Gruppenordnung unbekannt, im Allgemeinen nicht prim und nicht effizient berechenbar ist. Hierzu führen wir eine beliebige Primzahl q ein und reduzieren die Exponenten im Protokoll modulo q.

Das Protokoll

Schlüsselerzeugung: **A** führt die folgenden Schritte aus:

Wähle eine endliche Abelsche Gruppe G und eine zufällige 160-Bit-Primzahl q;
bestimme eine Approximation $L \approx |G|$ mit $L \geqslant |G|$;
wähle ein zufälliges Element $\gamma \in G$;
wähle eine Zufallszahl a aus $\{2, \ldots, L-1\}$ und berechne $\alpha = \gamma^{-a}$.

As öffentlicher Schlüssel ist (G, γ, α, q), **A**s privater Schlüssel ist a.

Identifikation: **A** wählt eine Zufallszahl r aus $\{1, \ldots, L-1\}$ *(Commitment)*;
A berechnet das Gruppenelement $\varrho = \gamma^r$ und sendet ϱ an **B** *(Zeuge, Witness)*;
B wählt ein zuvor nicht benutztes $e \in \{2, \ldots, q-1\}$ und sendet e an **A** *(Challenge)*;
A berechnet ganze Zahlen y, ℓ mit $ae + r = \ell q + y$ und $0 \leqslant y < q$;
A berechnet $\lambda = \gamma^\ell$;
A sendet (y, λ) *(Response)* an **B**.
B akzeptiert **A**s Identität genau dann, wenn $\gamma^y \alpha^e \lambda^q = \varrho$.

Das R-Schnorr-Identifikationsverfahren ist abgeschlossen (complete), wie der folgende Satz zeigt.

Satz 3.3.22
*Wenn **A** und **B** dem R-Schnorr-Identifikationsprotokoll folgen, dann ist die Verifikation immer erfolgreich.*

[7] genauer: äquivalent zum Faktorisieren der Diskriminante des Zahlkörpers

Beweis: Mit den Bezeichnungen von oben gilt $\gamma^y \alpha^e \lambda^q = \gamma^{ae+r} \cdot \gamma^{-\ell q} \cdot \gamma^{-ae} \cdot \gamma^{\ell q} = \gamma^r = \varrho$. □

Sicherheit

Die Sicherheit der von uns vorgestellten Variante R-Schnorr des Schnorr-Identifikationsverfahrens basiert auf dem Wurzelproblem (Root Problem). R-Schnorr steht für „Root-based Schnorr Identification Protocol".

Satz 3.3.23
*Ein Angreifer **E**, der in Polynomzeit das Wurzelproblem lösen kann, kann sich erfolgreich als beliebiger Teilnehmer ausgeben (impersonation attack).*

Beweis: Folgendermaßen gibt sich ein Angreifer **E**, der in Polynomzeit das Wurzelproblem lösen kann, erfolgreich als Teilnehmer **A** aus: **E** besorgt sich **A**s öffentlichen Schlüssel (G, γ, α, q), wählt $\tilde{\varrho} \in G$, $\tilde{y} \in \{2, \ldots, q-1\}$ beliebig und sendet $\tilde{\varrho}$ an den Verifizierer **B**. Der Angreifer **E** erhält von **B** die Challenge e und berechnet $\tau = \tilde{\varrho}\gamma^{-\tilde{y}}\alpha^{-e}$. Dann bestimmt **E** das Element $\tilde{\lambda}$ als q-te Wurzel von τ. Da q eine zufällig gewählte große Primzahl ist, teilt q die Gruppenordnung $|G|$ mit vernachlässigbar kleiner Wahrscheinlichkeit. Aber nur in diesem Fall kann es vorkommen, dass τ keine q-te Wurzel besitzt. Falls **E** keine q-te Wurzel aus τ ziehen kann, trifft er eine andere Wahl für $\tilde{\varrho}$ und $\tilde{y}$ und beginnt von vorne. Falls er $\tilde{\lambda}$ als q-te Wurzel aus τ bestimmen kann, sendet **E** das Paar $(\tilde{y}, \tilde{\lambda})$ als Response an **B**. Der Verifizierer **B** akzeptiert den Identitätsnachweis für **A**, denn es gilt $\tilde{\gamma}^{\tilde{y}} \alpha^e \tilde{\lambda}^q = \tilde{\gamma}^{\tilde{y}} \alpha^e \tau = \tilde{\varrho}$. □

Wir diskutieren weitere Aspekte der Sicherheit des R-Schnorr-Signaturverfahrens.

Betrugswahrscheinlichkeit. Das korrekte Erraten der Challenge e ermöglicht einem Angreifer **E**, sich erfolgreich als **A** auszugeben. **E** wählt hierzu beliebige $y \in \{0, \ldots, q-1\}$ und $\lambda \in G$, sendet dem Verifizierer $\varrho = \gamma^y \alpha^e \lambda^q$ als Zeugen und sendet (y, λ) als Response. Bei Anwendung dieses Angriffes hat **E** eine Erfolgschance von etwa 2^{-160}. Ein besserer Angriff ist uns nicht bekannt.

Soundness. Es ist nicht bekannt, ob R-Schnorr sound ist. Für das Originalverfahren [Sch90] kann die Soundness-Eigenschaft nachgewiesen werden (siehe [Sti95, Theorem 9.1]. Die Beweistechnik für das Originalverfahren lässt sich nicht auf R-Schnorr übertragen, da die Ordnung des Basiselementes γ in unserem Fall unbekannt und im Allgemeinen nicht prim ist.

Zero-Knowledge-Eigenschaft. R-Schnorr ist nicht Zero-Knowledge, da kein Polynomzeit-Simulator für den Beweiser **A** existiert. Denn ein Verifizierer **B** lernt bei Ausführung des Protokolls eine Lösung (ϱ, y, λ) der Gleichung $\varrho = \gamma^y \alpha^e \lambda^q$; eine solche Lösung hätte **B** ohne Interaktion mit dem Prover **A** nicht selbst bestimmen können, wenn die Challenge e beispielsweise in Abhängigkeit von ϱ gewählt wird. Das Original-Schnorr-Identifikationsverfahren ist ebenso wenig Zero-Knowledge (siehe [MOV97, Kap. 10.4.4]).

Sicherheit (Zusammenfassung). Es ist unbekannt, ob R-Schnorr unter plausiblen Annahmen sicher ist. Eine analoge Situation liegt für das Original-Schnorr-Identifikationsverfahren vor (siehe [Sti95, Kap. 9.2]). Das Originalverfahren ist genau wie unsere Modifikation R-Schnorr dennoch von praktischem Interesse, weil die Berechnungskomplexität für den Beweiser und die Kommunikationslast gering sind (siehe unten).

Effizienz

Wir diskutieren die Effizienz des R-Schnorr-Identifikationsverfahrens.

Berechnungskomplexität R-Schnorr erfordert vom Beweiser **A** zwei Exponentiationen von Gruppenelementen. Hiervon kann eine Exponentiation (mit Exponent in der Größenordnung der Gruppenordnung) komplett vorberechnet werden; die andere Exponentiation (mit Exponent $\approx 2^{160}$) muss zur Laufzeit durchgeführt werden, wobei Vorberechnungstechniken eingesetzt werden können. Der Verifizierer **B** hat eine simultane Exponentiation (mit 160-Bit-Exponent) und einen Gleichheitstest durchzuführen.

Speicherplatzbedarf für Geheimnisse Die Teilnehmer benötigen zur Speicherung ihrer R-Schnorr-Geheimnisse (private Schlüssel) Speicherplatz für eine positive ganze Zahl in der Größenordnung der Gruppenordnung. Unsere Analyse aus Kapitel 4 zeigt: Bei Wahl eines Stender-Körpers vom Grad 3 mit Diskriminante Δ in der Größenordnung von 2^{-182} ist das Lösen eines diskreten Logarithmusproblems oder eines Wurzelproblems nach heutigem Kenntnisstand ebenso schwer wie das Faktorisieren eines RSA 1024-Bit-Modulus. Bei der genannten Wahl des Stender-Körpers erwarten wir die Gruppenordnung (Klassenzahl) in der Größenordnung $|\Delta|^{0.45} \sim 2^{182}$. Zur Speicherung des R-Schnorr-Geheimnisses benötigt ein Teilnehmer im Beispiel der genannten Wahl der Klassengruppe eines Stender-Körpers also 182 Bits.

Kommunikationslast Im R-Schnorr-Protokoll werden zwei 160-Bit-Zahlen und zwei Gruppenelemente übertragen. Die Kommunikationslast kann reduziert werden, indem der Beweiser **A** als Zeugen statt des Gruppenelementes ϱ einen Hashwert von ϱ schickt. Die Hashfunktion wird in diesem Fall vorher für alle Teilnehmer festgelegt und veröffentlicht; zur Verifikation berechnet der Verifizierer $\varrho' = \gamma^y \alpha^e \lambda^q$ und prüft, ob $h(\varrho) = h(\varrho')$ gilt.

Empfehlung zur Parameterwahl

R-Schnorr ist ein interaktives Identifikationsverfahren, bei welchem dem Beweiser zufällige Challenges vorgelegt werden. Für geeignete Gruppen G ist kein besserer Angriff bekannt, als die Challenges in jeder Runde zu raten. Um Offline-Attacken auszuschließen, empfehlen wir die Wahl einer Primzahl q in der Größenordnung 2^{160} (siehe oben). Hiermit erhalten wir ein Sicherheitsniveau von 2^{-160}.

Gemäß der Meinung von anerkannten Experten (siehe z.B. [FS87], [MOV97, Kap. 10.4.2]) genügt ein Sicherheitsniveau von 2^{-20} in der Praxis für fast alle Anwendungen; denn Identifikationssysteme sind in der Regel so konstruiert, dass eine kleine Anzahl an Fehlversuchen unter dem Namen eines Teilnehmers zum (temporären) Sperren des Teilnehmers führt. Daher empfehlen wir, die Challenge e im R-Schnorr-Identifikationsverfahren zur Reduzierung der Berechnungskomplexität wie folgt zu wählen: $1 < e \leqslant 2^t < q$, wobei $t \in \mathbb{Z}_{>0}$, z.B. $t = 20$). Hierdurch reduzieren wir das Sicherheitsniveau auf 2^{-t} (also z.B. auf 2^{-20}).

Weitere Identifikationsprotokolle

In endlichen Abelschen Gruppen, die den Anforderungen aus Abschnitt 4.2 genügen, (also z.B. in den in Abschnitt 4.3 vorgestellten Klassengruppen algebraischer Zahlkörper) lässt sich auch

die die Ohta-Okamoto-Variante [OO90, OO88] des Guillou-Quisquater-Identifikationsverfahrens implementieren. Diese haben wir implizit in Abschnitt 3.3.5.2 vorgestellt und zur Konstruktion des Signaturverfahrens DC-GQ mit designiertem Signierer verwendet.

3.3.5.5 Bit Commitment und fairer Münzwurf

Wir stellen in diesem Abschnitt zwei neue Bit Commitment-Verfahren vor. Wir zeigen, dass die Verfahren sicher in Gruppen sind, welche den Anforderungen aus Abschnitt 4.2 genügen.

Bit Commitment basierend auf dem Wurzelproblem

Setup: Die Teilnehmer einigen sich auf eine endliche Abelsche Gruppe G und eine hinreichend große Primzahl p mit $2^{t-1} < p \leqslant 2^t$ für einen Sicherheitsparameter $t \in \mathbb{Z}_{>0}$ (man kann in Analogie zur Wahl der Untergruppenordnung im Original-DSA-Verfahren [18694] $t = 160$ wählen);
B wählt ein zufälliges Gruppenelement $\mu \in G - \{1\}$ und sendet μ an **A**.

Commitment: **A** überprüft, dass $\mu \in G - \{1\}$;
A wählt $b \in \{0, 1\}$, wählt ein zufälliges Gruppenelement $\xi \in G - \{1\}$ und berechnet den Blob $\eta = BC_\mu(b, \xi) = \mu^b \xi^p$;
A sendet η an **B**.

Öffnen des Blobs: **A** veröffentlicht $b \in \{0, 1\}$ und $\xi \in G - \{1\}$.

Verifikation: **B** verifiziert, dass $\eta = \mu^b \xi^p$.

Satz 3.3.24
Obiges Bit Commitment-Verfahren ist sicher, falls es probabilistisch in Polynomzeit unmöglich ist, folgendes zu tun:

1. *Falls $p \nmid |G|$, löse ein spezielles p-tes Wurzelproblem.*
2. *Falls $p \mid |G|$, entscheide, ob zu vorgegebenem $\alpha \in G$ eine p-te Wurzel $\xi \in G$ existiert.*

Beweis:

Verbindlichkeit (binding). Offensichtlich kann **A** den Blob später durch Veröffentlichung von ξ öffnen. Angenommen, ein Betrüger **A** könnte einen Blob sowohl zu $b = 1$ als auch zu $b = 0$ öffnen. Dann müsste **A** Gruppenelemente $\xi_1, \xi_2 \in G - \{1\}$ kennen mit $\mu \xi_1^p = \xi_2^p$. Der Betrüger könnte also zu vorgegebenem beliebigem Element $\mu \in G - \{1\}$ mit $\xi_2 \xi_1^{-1}$ eine p-te Wurzel berechnen, im Widerspruch zur Voraussetzung.

Geheimhaltung (concealing). Angenommen, ein Betrüger **B** könne im Protokoll ein Gruppenelement $\mu \in G - \{1\}$ so wählen, dass er anschließend einen Blob η öffnen kann. Dann kann **B** zu vorgegebenem $\eta \in G$ entscheiden, ob ein $\xi \in G - \{1\}$ existiert mit $\eta = \mu \xi^p$ ($\Leftrightarrow \eta \mu^{-1} = \xi^p$) oder ob ein solches ξ existiert mit $\eta = \xi^p$.
Fall 1: **B** wählte μ so (evtl. ohne sich dessen bewusst zu sein), dass für alle $\upsilon \in G$ gilt $\mu \neq \upsilon^p$. Hieraus folgt $p \mid |G|$. Also kann **B** zu vorgegebenem $\eta \in G$ entscheiden, ob η eine p-te Wurzel in G besitzt. Widerspruch zur Voraussetzung.

Fall 2: **B** wählte μ so (evtl. ohne sich dessen bewusst zu sein), dass ein $\upsilon \in G$ existiert mit $\mu = \upsilon^p$. Also kann **B** zu vorgegebenem $\eta \in G$ entscheiden, ob **A** die Wahl $b = 1$ getroffen hat (d.h. $\eta = (\upsilon\xi)^p$ für ein $\upsilon \in G$) oder ob **A** sich für $b = 0$ entschieden hat (d.h. $\eta = \xi^p$). Dies zu entschieden ist **B** aber unmöglich, da **A** ξ zufällig in $G - \{1\}$ gewählt hat und für fixes ξ und variables $\upsilon \in G$ alle Gruppenelemente durchlaufen werden.

□

Bit Commitment basierend auf dem diskreten Logarithmusproblem

Setup: Die Teilnehmer einigen sich auf eine endliche Abelsche Gruppe G, eine Approximation L der Gruppenordnung $|G|$ und ein zufälliges Gruppenelement $\alpha \in G - \{1\}$;
B wählt ein zufälliges Gruppenelement $\mu \in G - \{1\}$ und sendet μ an **A**.

Commitment: **A** überprüft, dass $\mu \in G - \{1\}$;
A wählt $b \in \{0, 1\}$, wählt ein zufälliges $x \in \{1, \ldots, L - 1\}$ und berechnet den Blob $\eta = BC_\mu(b, x) = \mu^b \alpha^x$;
A sendet η an **B**.

Öffnen des Blobs: **A** veröffentlicht $b \in \{0, 1\}$ und $x \in \{1, \ldots, L - 1\}$.

Verifikation: **B** verifiziert, dass $\eta = \mu^b \alpha^x$.

Satz 3.3.25
Obiges Bit Commitment-Verfahren ist sicher, falls es probabilistisch in Polynomzeit unmöglich ist, folgendes zu tun:

1. *Falls G zyklisch ist, berechne einen diskreten Logarithmus in G.*
2. *Falls G nicht zyklisch ist, entscheide, ob zu vorgegebenen Elementen $\alpha, \beta \in G$ der diskrete Logarithmus $\log_\alpha \beta$ existiert.*

Beweis:

Verbindlichkeit (binding). Offensichtlich kann **A** den Blob später durch Veröffentlichung von x öffnen. Angenommen, ein Betrüger **A** könnte einen Blob sowohl zu $b = 1$ als auch zu $b = 0$ öffnen. Dann müsste **A** zu beliebigem $\mu \in G - \{1\}$ Zahlen $x_1, x_2 \in \{1, \ldots, L - 1\}$ kennen mit $\mu\alpha^{x_1} = \alpha^{x_2}$. Wegen $\mu = \alpha^{x_2 - x_1}$ könnte der Betrüger also zu vorgegebenem beliebigem Element $\mu \in G - \{1\}$ den diskreten Logarithmus $\log_\alpha \mu$ berechnen, im Widerspruch zur Voraussetzung.

Geheimhaltung (concealing). Angenommen, ein Betrüger **B** könne im Protokoll ein Gruppenelement $\mu \in G - \{1\}$ so wählen, dass er anschließend einen Blob η öffnen kann. Dann kann **B** zu vorgegebenem $\eta \in G$ entscheiden, ob eine positive ganze Zahl x existiert mit $\eta = \mu\alpha^x$ ($\Leftrightarrow \eta\mu^{-1} = \alpha^x$) oder ob ein solches x existiert mit $\eta = \alpha^x$.
Fall 1: **B** wählte $\mu \notin \langle\alpha\rangle$ (evtl. ohne sich dessen bewusst zu sein). Also kann **B** zu vorgegebenem $\eta \in G$ entscheiden, ob $\eta \in \langle\alpha\rangle$ oder $\eta \notin \langle\alpha\rangle$, im Widerspruch zur Voraussetzung.
Fall 2: **B** wählte $\mu \in \langle\alpha\rangle$ (evtl. ohne sich dessen bewusst zu sein). Also kann **B** zu vorgegebenem $\eta \in G$ entscheiden, ob **A** die Wahl $b = 1$ getroffen hat (d.h. $\eta = \alpha^{y+x}$ für eine positive ganze Zahl y) oder ob **A** sich für $b = 0$ entschieden hat (d.h. $\eta = \alpha^x$). Dies zu entschieden ist **B** aber unmöglich, da **A** die Zahl x zufällig in $\{1, \ldots, L - 1\}$ gewählt hat.

□

Optimierung der Verfahren Stinson [Sti95, Kap. 13] verfeinert den Begriff der Geheimhaltung des Blobs wie folgt: Ein Bit Commitment-Verfahren heißt *computational concealing (berechenbar geheimhaltend)*, wenn kein Angreifer das Bit b probabilistisch in Polynomzeit aus dem Blob $BC(b, x)$ berechnen kann. Ein Bit Commitment-Verfahren heißt *unconditional concealing (uneingeschränkt geheimhaltend)*, wenn kein Angreifer mit unbeschränkten Ressourcen das Bit b aus dem Blob $BC(b, x)$ berechnen kann.

Beide von uns vorgestellte Bit Commitment-Verfahren sind computational concealing. Das erste der beiden Bit Commitment-Verfahren wird unconditional concealing, wenn ein Trust Center jeweils ein zufälliges Gruppenelement μ vorgibt, welches p-te Potenz in G ist. Das zweite Bit Commitment-Verfahren wird unconditional concealing, wenn ein Trust Center jeweils ein zufälliges Gruppenelement μ vorgibt, welches in der von α erzeugten Untergruppe in G ist.

Fairer Münzwurf

Mittels Bit Commitment-Verfahren lässt sich ein *fairer Münzwurf* realisieren, bei dem die beiden Teilnehmer keine Sichtverbindung zueinander haben. Hierbei legt sich **A** auf ein Bit b fest und übergibt an **B** den generierten Blob η. **B** rät nun den Wert von b, und **A** öffnet den Blob, um b offenzulegen. Hat **B** richtig geraten, ist das Resultat des virtuellen Münzwurfs 1, andernfalls 0. Auf die gleiche Art und Weise können zwei Parteien einen gemeinsamen zufälligen Schlüssel aushandeln, ohne dass eine von beiden Parteien Einfluss auf den Schlüssel hat: Für jedes Schlüsselbit wird ein fairer Münzwurf gemacht.

Kapitel 4

Auswahl kryptographisch geeigneter Klassengruppen

Die NF-Kryptoverfahren sollen in der Praxis effizient sein und ein gewünschtes messbares Sicherheitslevel übersteigen. Als Maßstab für das Sicherheitslevel sollen uns dabei in der Praxis gängige Public-Key-Systeme dienen wie z.B. RSA-1024 Bit. In diesem Kapitel erläutern wir, wie Klassengruppen algebraischer Zahlkörper ausgewählt werden sollen, um die beiden genannten Ziele zu erreichen.

Bislang war vollkommen unklar, wie für NF-Kryptoverfahren die Schlüssellänge, d.h. die Diskriminante und der Grad eines algebraischen Zahlkörpers, gewählt werden soll, um ein gewünschtes Sicherheitsniveau zu erhalten. Mit längeren Schlüsseln wird es schwieriger, das zugrunde liegende mathematische Problem zu lösen, um das jeweilige NF-Kryptoverfahren zu brechen. Zur Analyse, welche Schlüssellängen zu wählen sind, muss man die Laufzeit des schnellsten Verfahrens kennen, welche das jeweilige NF-Kryptoverfahren bricht. Dazu sind keine schnelleren Methoden bekannt als das zugrunde liegende mathematische Problem zu lösen. Wir werden sehen: Die schnellsten Algorithmen zum Lösen des Wurzelproblems, des diskreten Logarithmusproblems und des Diffie-Hellman-Problems in Klassengruppen algebraischer Zahlkörper sind Varianten von Buchmanns Klassengruppenalgorithmus ([Buc89], siehe auch [Coh95]).

In diesem Kapitel schließen wir diese Lücke und erklären, wie Schlüssellängen in NF-Kryptoverfahren zu wählen sind, um ein gewünschtes Sicherheitslevel zu erreichen. Dabei wenden wir das Modell [LV01] von Lenstra und Verheul an. Somit können wir abschätzen, welche Schlüssellängen in der Zukunft zu wählen sind, um ein Sicherheitsniveau zu erhalten, welches dem traditioneller Kryptosysteme bei den in [LV01] angegebenen Schlüssellängen entspricht.

Wir werden sehen, dass Diskriminanten der Länge 404 Bits (550 Bits) in kubischen Stender-Körpern nach heutigem Kenntnisstand die gleiche Sicherheit bieten wie RSA-Moduli der Länge 1024 Bits (2048 Bits). Der schnellste Algorithmus zum Brechen der Kryptoverfahren über imaginär-quadratischen Zahlkörpern hat bekanntermaßen die Komplexität $L(|\Delta|)[\frac{1}{2}, 1 + o(1)] = e^{(1+o(1))\ln(|\Delta|)^{\frac{1}{2}}\ln(\ln(|\Delta|))^{\frac{1}{2}}}$ ([HM00]). Daher entsprechen obige Diskriminantenlängen 687 Bit-Diskrimanten (1208 Bit-Diskrimanten) eines imaginär-quadratischen Zahlkörpers ([HM00]).

Wir gehen in diesem Kapitel folgendermaßen vor:

Wir beschreiben die nach heutigem Kenntnisstand denkbaren Angriffe auf die NF-Kryptoverfahren (Abschnitte 4.1.1 bis 4.1.6). Hierbei nehmen wir an, die Kryptoverfahren

könnten nur durch Lösen des jeweils zugrunde liegenden algorithmischen Problems gebrochen werden. Wir geben die Komplexität der Angriffe an und berichten, sofern vorhanden, über praktische Erfahrungen mit deren Effizienz. Insbesondere untersuchen wir die Laufzeit von Buchmanns Klassengruppenalgorithmus und verbessern bisherige Komplexitätsresultate.

Nun analysieren wir, wie groß die in den Komplexitäten der Angriffe führenden Terme sein müssen, damit die jeweiligen Angriffe in der Praxis zu ineffizient sind. Hieraus folgen notwendige Bedingungen für kryptographisch geeignete Klassengruppen algebraischer Zahlkörper (Abschnitt 4.2).

Dann erklären wir, wie man Zahlkörper und Klassengruppen erzeugt, die diesen Bedingungen genügen (Abschnitt 4.3). Insbesondere müssen die Zahlkörper einen kleinen Regulator und eine große Klassenzahl besitzen. Hier sind wir mit dem Problem konfrontiert, dass sich Zahlkörper vom Grad > 2 normalerweise genau umgekehrt verhalten (großer Regulator und kleine Klassenzahl). Dennoch gelingt es uns, Kandidaten für kryptographisch geeignete Klassengruppen vorzustellen und im anschließenden Abschnitt 4.4 deren kryptographische Eignung nachzuweisen.

Zur weiteren Analyse ist es wichtig, den nach heutigem Kenntnisstand schnellsten Algorithmus zu identifizieren, welcher das Kryptosystem durch Lösen des zugrunde liegenden algorithmischen Problems bricht. Zudem muss eine die Laufzeit dieses Algorithmus approximierende Funktion bekannt sein. In der Praxis gibt es häufig nicht *einen einzigen* Algorithmus, der *alle* Instanzen eines algorithmischen Problems effizient löst. Besitzt ein algebraischer Zahlkörper beispielsweise eine große Diskriminante und eine glatte Klassenzahl, so löst unser (p-1)-Algorithmus aus Abschnitt 4.1.5 am schnellsten ein beliebiges Wurzelproblem in der betreffenden Klassengruppe; bei großer Diskriminante und großem Primteiler in der Klassenzahl ist aber Buchmanns Algorithmus am schnellsten. Wir werden dennoch Buchmanns Algorithmus als für alle zugelassenen Situationen schnellsten Algorithmus identifizieren, da wir durch geschickte Wahl der algebraischen Zahlkörper Randfälle ausschließen können, in denen andere Algorithmen effizienter als Buchmanns Algorithmus sind.
(Die Identifikation des schnellsten Algorithmus erfolgt in Abschnitt 4.1, der Ausschluss der Randfälle in den Abschnitten 4.1 bis 4.4.)

Schließlich zeigen wir in Abschnitt 4.5, wie die Schlüssellänge (Diskriminante) eines algebraischen Zahlkörpers in Abhängigkeit vom Zahlkörpergrad gewählt werden muss, um für ein NF-Kryptoverfahren ein gewünschtes Sicherheitslevel zu erhalten. Wir gehen aus von Laufzeitmessungen des effizientesten Algorithmus zum Brechen der NF-Kryptosysteme (Buchmanns Klassengruppenalgorithmus) für betragsmäßig künstlich kleine Diskriminanten. Mit Kenntnis der von uns in Abschnitt 4.1.6 bewiesenen approximierenden Laufzeitfunktion erhalten wir durch Extrapolation die erwartete Laufzeit für betragsmäßig größere Diskriminanten (die für kryptographische Anwendungen realistischer sind). Die auf größere Eingabeparameter extrapolierten Laufzeiten entsprechen gewissen Sicherheitsniveaus für die NF-Kryptosysteme. Dann wenden wir das Modell [LV01] von Lenstra und Verheul an, um unsere Schlüssellängen mit entsprechenden Schlüssellängen von RSA, ECDSA und mit den entsprechenden Schlüssellängen der Kryptoverfahren über imaginärquadratischen Zahlkörpern zu vergleichen. Dabei entspricht die Schlüssellänge eines anderen Kryptosystems einer Schlüssellänge unserer NF-Kryptosysteme, wenn der gleiche Berechnungsaufwand (d.h. die gleiche Anzahl an MIPS-Jahren) notwendig ist, um beide Kryptosysteme zu brechen. Wir sagen dann auch, die Kryptosysteme böten die gleiche Sicherheit und sprechen von *sicherheitstechnisch äquivalenten Schlüssellängen*.

In diesem Kapitel ist der folgende Begriff von Bedeutung:

Definition 4.0.26
*Sei $B \in \mathbb{R}_{>0}$. Wir nennen eine ganze Zahl B-*glatt*, wenn sie ausschließlich Primteiler kleiner oder gleich B besitzt.*

4.1 Angriffe auf die NF-Kryptoverfahren

Ziel dieses Abschnittes ist es, den in der Praxis effizientesten Algorithmus zum Brechen der NF-Kryptoverfahren aus Kapitel 3 zu identifizieren. Wir betrachten nur Angriffe durch Lösen der zugrunde liegenden algorithmischen Probleme, also des Wurzelproblems, des diskreten Logarithmusproblems und des Diffie-Hellman-Problems in Klassengruppen algebraischer Zahlkörper.

In Abschnitt 3.1 haben wir gezeigt, dass man das Wurzelproblem in der Praxis durch Lösen des Klassenzahlproblems und anschließender Anwendung einer Polynomzeitreduktion löst. Das Klassenzahlproblem kann man durch Lösen eines diskreten Logarithmusproblems in der Klassengruppe lösen. Auch das Diffie-Hellman-Problem löst man durch Berechnung eines diskreten Logarithmus.

Daher stellen wir die bekannten (mehr oder minder effizienten) Algorithmen zur Berechnung der Klassenzahl und diskreter Logarithmen in Klassengruppen algebraischer Zahlkörper vor. Sei G eine endliche Abelsche Gruppe, und seien $\alpha, \beta \in G$ beliebige Gruppenelemente. Dann beziehen wir uns im Folgenden beim diskreten Logarithmusproblem darauf, eine ganzzahlige Lösung x der folgenden Gleichung zu finden:

$$\alpha^x = \beta \tag{4.1}$$

Wir unterscheiden dabei zwischen generischen Algorithmen und nicht-generischen Algorithmen. Einen Algorithmus, der auf einer endlichen Abelschen Gruppe G operiert, bezeichnen wir als *generisch*, wenn er keine speziellen Eigenschaften der Gruppe ausnutzt und in allen Gruppen anwendbar ist, in denen sich folgende Operationen für beliebige Gruppenelemente $\alpha, \beta \in G$ ausführen lassen:

1. die Gruppenoperation $\alpha \cdot \beta$
2. Berechnung des Inversen α^{-1}
3. Gleichheitsentscheid $\alpha \stackrel{?}{=} \beta$

Wir skizzieren jeweils die Kernidee der Algorithmen und geben deren Komplexität an. Eine Sonderrolle kommt hier dem Klassengruppen-spezifischen (d.h. nicht-generischen) Buchmann-Algorithmus zu. Denn dessen Komplexität und praktische Effizienz bei Anwendung in den von uns vorgeschlagenen Klassengruppen ist bis heute unbekannt. Wir analysieren Buchmanns Algorithmus daher detailliert in Abschnitt 4.1.6. Tatsächlich erweist sich Buchmanns Algorithmus in unserem Kontext in der Praxis als effizienter als alle anderen Algorithmen.

Als Referenzkomplexität für ein schwieriges algorithmisches Problem betrachten wir den Aufwand zur Faktorisierung einer RSA-1024-Bit-Zahl mit dem schnellsten bekannten Algorithmus, dem General Number Field Sieve (GNFS). Hier müssen etwa $6.01 \cdot 10^{10}$ MIPS-Jahre aufgebracht werden. In Klassengruppen kubischer Zahlkörper entspricht dies dem Durchführen von etwa 2^{57} Gruppenoperationen[1]. Wenn wir nämlich die plausible Annahme machen, dass ein 400 MIPS-Rechner 25 msec zum Durchführen einer Gruppenoperation (Idealmultiplikation und Reduktion) benötigt, so führt der 400 MIPS-Rechner bis zu 40 Gruppenoperationen in einer Sekunde aus. Für

[1] In Zahlkörpern höherer Grade entspricht dies weniger als 2^{57} Gruppenoperationen.

1 MIPS-Jahr viele Bitoperationen benötigt der 400 MIPS-Rechner $\frac{1}{400} \cdot (60 \cdot 60 \cdot 24 \cdot 365)$ sec. Folglich entsprechen auf einem 400 MIPS-Rechner $6.01 \cdot 10^{10}$ MIPS-Jahre folgender Anzahl an Gruppenoperationen in Klassengruppen algebraischer Zahlkörper:

$$6.01 \cdot 10^{10} \cdot \frac{1}{400} \cdot (60 \cdot 60 \cdot 24 \cdot 365) \cdot 40 \approx 2^{57} =: \mathrm{A}_{total}$$

4.1.1 Baby-Step-Giant-Step-Algorithmus

Der Algorithmus Wir diskutieren zunächst den generischen Baby-Step-Giant-Step-Algorithmus von Shanks [Sha71]. Die Kenntnis der Gruppenordnung $|G|$ wird nicht benötigt; statt dessen kann mit einer groben Approximation von $|G|$ gearbeitet werden. (Die Variante von Terr [Ter99] kommt sogar ohne Kenntnis der Größenordnung von $|G|$ aus.) Das diskrete Logarithmusproblem (4.1) wird hier wie folgt gelöst: Sei $m = \lfloor \sqrt{|G|} \rfloor + 1$, dann existieren $x_1, x_2 \in \{0, \ldots, m\}$ mit $x = x_1 m + x_2$, $x_2 \neq m$. Wenn $\alpha^x = \beta$ ist, dann gilt $\alpha^{x_1 m} = \alpha^{-x_2} \beta$. Zunächst wird eine Menge $M = \{\alpha^{mi} \mid 0 \leqslant i \leqslant m\}$ berechnet (die giant steps). Dann wird geprüft, ob eines der Elemente $\alpha^{-j}\beta$ für $j \in \{0, \ldots, m-1\}$ in M liegt (die baby steps). Falls dies der Fall ist, so ergibt sich wegen $\alpha^{mi} = \alpha^{-j}\beta$ die Lösung $x = mi + j$. Falls keins der Elemente $\alpha^{-j}\beta$ in M liegt, so hat (4.1) keine Lösung.

Komplexität Shanks Baby-Step-Giant-Step-Algorithmus ist deterministisch und hat Zeit- und Platzkomplexität $O(\sqrt{|G|})$ mit minimaler O-Konstante > 1. Verfeinerungen des Algorithmus und detaillierte Komplexitätsanalysen finden sich beispielsweise in [BJT97]. Dort wird auch über praktische Erfahrungen bei der Implementierung des Algorithmus berichtet.

Schlussfolgerung Das Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve ist so aufwändig wie das Durchführen von 2^{57} Gruppenoperationen in der Klassengruppe eines kubischen Stender-Zahlkörpers (siehe oben). Der Baby-Step-Giant-Step-Algorithmus benötigt mindestens 2^{57} viele Gruppenoperationen, wenn $\operatorname{ord} \alpha \geqslant 2^{114}$. Damit der Baby-Step-Giant-Step-Algorithmus in der Praxis nicht erfolgreich durchführbar ist, müssen wir dafür Sorge tragen, dass die Ordnung $\operatorname{ord} \alpha$ eines zufällig gewählten Elementes $\alpha \in G$ hinreichend groß ist. In Abschnitt 4.2 werden wir zeigen, dass wir daher $|\Delta| \geqslant 2^{253}$ fordern müssen.

4.1.2 Pohlig-Hellman-Algorithmus

Der Algorithmus Der Pohlig-Hellman-Algorithmus [PH78] löst das Problem (4.1), falls eine Faktorisierung der Gruppenordnung in paarweise teilerfremde Zahlen gegeben ist: $|G| = \prod_{i=1}^{k} m_i$, $ggT(m_i, m_j) = 1$ für $i \neq j$. Durch Potenzierung beider Seiten der Gleichung (4.1) zur Potenz $\frac{m}{m_i}$ projiziert man das Problem (4.1) für $i = 1, \ldots, k$ in eine Untergruppe G_i von G der Ordnung m_i. Für das so entstehende DL-Problem in G_i finde man eine Lösung $x_i \mod m_i$. Hierzu wird ein beliebiger Algorithmus angewendet, beispielsweise der Baby-Step-Giant-Step-Algorithmus aus Abschnitt 4.1.1. Die so erhaltenen Teillösungen $x_i \mod m_i$ setzt man mit dem Chinesischen Restsatz (siehe z.B. [Coh95]) zu einer Lösung $x \mod |G|$ von (4.1) zusammen.

Komplexität Bei bekannter Faktorisierung $|G| = \prod_{i=1}^{k} m_i$ und größtem Primteiler p von $|G|$ muss man bei Kombination des Pohlig-Hellman-Algorithmus mit dem Baby-Step-Giant-Step-Algorithmus mindestens $\sqrt{p}$ viele Gruppenoperationen ausführen.

Schlussfolgerung Der Pohlig-Hellman-Algorithmus erfordert die Kenntnis der Primfaktorzerlegung der Gruppenordnung bzw. der Elementordnung von α oder eines kleinen Vielfachen. Der Erfolg des Pohlig-Hellman-Algorithmus wird in der Praxis verhindert, wenn die genannten Ordnungen (oder ein kleines Vielfaches) nicht berechnet werden können. Hierzu muss bei Verwendung von Klassengruppen algebraischer Zahlkörper die Diskriminante dem Betrage nach groß sein.

4.1.3 Pollards Algorithmen

Der Algorithmus Pollards Rho-Algorithmus [Pol78] setzt die exakte Kenntnis der Gruppenordnung voraus. Somit ist der Algorithmus in den von uns in Abschnitt 4.3 vorgeschlagenen Klassengruppen nicht anwendbar. (Bei bekannter Gruppenordnung ist die erwartete Laufzeit zur Lösung von (4.1) $O(\sqrt{|G|})$ viele Gruppenoperationen bei konstantem Speicherplatzbedarf.)

Pollards Lambda-Algorithmus [Pol78] kann nur benutzt werden, wenn bekannt ist, dass eine Lösung x von (4.1) existiert und in einem spezifizierten Intervall der Länge w liegt.

Komplexität Bei bekanntem Lösungsintervall der Länge w benötigt Pollards Lambda-Algorithmus $O(\sqrt{w})$ viele Gruppenoperationen und in der Relation hierzu wenig Speicherplatz. Der Lambda-Algorithmus ist vom Typ Monte Carlo, d.h. er gibt mit einer gewissen Fehlerwahrscheinlichkeit ein falsches Resultat aus. Durch Verwendung des spezifizierten Intervalls $\{1, \ldots, L\}$ mit $L \approx |G|$ kann der Lambda-Algorithmus auch angewendet werden, wenn lediglich eine Approximation L für die Gruppenordnung, nicht aber ein kleineres Lösungsintervall bekannt ist. Der Lambda-Algorithmus benötigt dann etwa $O(\sqrt{|G|})$ viele Gruppenoperationen.

Schlussfolgerung In unserem Kontext – der Klassengruppe $G = Cl(\Delta)$ eines algebraischen Zahlkörpers – ist die Gruppenordnung $|G|$ unbekannt. Den Lambda-Algorithmus können wir in der Untergruppe $\langle\alpha\rangle$ der Gruppe G anwenden. Ist $\operatorname{ord}\alpha \leqslant L$ für eine Schranke L, so benötigt Pollards Lambda-Algorithmus etwa $O(\sqrt{L})$ viele Gruppenoperationen. Um den Erfolg von Pollards Lambda-Algorithmus in der Praxis zu verhindern, muss $\operatorname{ord}\alpha$ groß sein. Das Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve ist so aufwändig wie das Durchführen von 2^{57} Gruppenoperationen in der Klassengruppe eines kubischen Stender-Zahlkörpers (siehe oben). Pollards Lambda-Algorithmus benötigt mindestens 2^{57} viele Gruppenoperationen, wenn $\operatorname{ord}\alpha \geqslant 2^{114}$, wenn also $|\Delta| \geqslant 2^{253}$ (siehe Abschnitt 4.2).

4.1.4 Index-Calculus-Algorithmus

Der Algorithmus Index-Calculus-Methoden nutzen in gewisser Weise den Hauptsatz über endliche Abelsche Gruppen aus: Für jede endliche Abelsche Gruppe G existieren Zahlen $m_1, \ldots, m_k \in \mathbb{Z}_{\geqslant 2}$, $k \in \mathbb{Z}_{\geqslant 1}$ mit $m_i \mid m_{i+1}$ und

$$G \cong \mathbb{Z}/m_1\mathbb{Z} \times \ldots \times \mathbb{Z}/m_k\mathbb{Z} \tag{4.2}$$

Index-Calculus-Methoden faktorisieren auf der linken Seite von (4.2) und nutzen Methoden der Linearen Algebra auf der rechten Seite von (4.2). Index-Calculus-Methoden bilden die

Faktorgruppe, die von in gewisser Weise primen Elementen A erzeugt wird modulo einer Menge B von multiplikativen Identitäten: $\langle A\rangle/B$. Wenn G durch eine solche Faktorgruppe dargestellt ist, führen Faktorisierungen in A zu Identitäten in G. Auf diese Weise können durch Anwendung von Methoden der Linearen Algebra die m_i in (4.2) und diskrete Logarithmen in G berechnet werden.

Komplexität Index-Calculus-Algorithmen haben in der Regel eine in der binären Länge der Diskriminante subexponentielle Laufzeit. Der effizienteste heute bekannte Index-Calculus-Algorithmus für Klassengruppen algebraischer Zahlkörper ist der Algorithmus von Buchmann [Buc89]. Über dessen praktische Laufzeit ist wenig bekannt. In Abschnitt 4.1.6 stellen wir Buchmanns Algorithmus vor und untersuchen dessen praktisches Laufzeitverhalten.

Schlussfolgerung Zur Verhinderung des Erfolges von Index-Calculus-Methoden muss bei Verwendung von Klassengruppen algebraischer Zahlkörper die Diskriminante dem Betrage nach hinreichend groß gewählt werden. Insgesamt stellt sich heraus, dass bei Verwendung kryptographisch geeigneter Klassengruppen algebraischer Zahlkörper die Anwendung von Buchmanns Algorithmus im Allgemeinen für einen Angreifer am Aussichtsreichsten erscheint. Das Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve ist so aufwändig wie das Durchführen von 2^{57} Gruppenoperationen in der Klassengruppe eines kubischen Stender-Zahlkörpers (siehe oben). In Abschnitt 4.5 werden wir sehen: Buchmanns Algorithmus benötigt hier mindestens 2^{57} viele Gruppenoperationen, wenn $|\Delta| \geqslant 2^{404}$ gilt.

4.1.5 (p-1)-Algorithmus

Wir analysieren nun einen Algorithmus, der erfolgversprechend scheint, wenn die Gruppenordnung $|G|$ glatt ist. Zuvor betrachten wir zunächst den Fall, $|G|$ enthalte einen großen glatten Faktor M. Wenn M bekannt ist, können diskrete Logarithmen in $\{\pm 1, \ldots, \pm M\}$ mit dem Pohlig-Hellman-Algorithmus effizient in G berechnet werden ([OW96]). Allerdings ist kein Algorithmus bekannt, der M berechnet ohne $|G|$ zu kennen. Die Berechnung von $|G|$ ist in den von uns vorgeschlagenen Klassengruppen algebraischer Zahlkörper aber in der Praxis nicht möglich.

Der Algorithmus Unser (p-1)-Algorithmus arbeitet ähnlich wie der (p-1)-Faktorisierungsalgorithmus [Pol75]. Angenommen, $|G|$ ist s-glatt mit $|G| \leqslant B$. Es sei $k := \prod_{p \leqslant s} p^{t_p}$, p prim, t_p maximal, so dass $p^{t_p} \leqslant B$. Dann ist für $\alpha \in G$ beliebig $\alpha^k = 1$.

Erster Schritt: In einem ersten Schritt bestimmt der Algorithmus folgendermaßen ein Vielfaches von $\operatorname{ord}\alpha$: Der Algorithmus ermittelt eine obere Schranke B der Gruppenordnung $|G|$; er rät ein geeignetes s und unterstellt die s-Glattheit von $|G|$; er berechnet wie oben dargestellt k und testet, ob $\alpha^k = 1$.

Zweiter Schritt: Wenn ein k gefunden wurde mit $\alpha^k = 1$, wird ein Wurzelproblem durch Berechnung eines erweiterten ggT und anschließende Exponentiation bzw. ein diskretes Logarithmusproblem durch Anwendung des Pohlig-Hellman-Algorithmus gelöst. Andernfalls wird ein größerer Wert für s geraten und das Verfahren wiederholt.

Komplexität Unter der Annahme, eine geeignete obere Schranke B der Gruppenordnung sei bekannt, benötigt allein der erste Schritt unseres (p-1)-Algorithmus $c \cdot s \cdot \frac{\log B}{\log s}$ viele Gruppenoperationen mit $c > 0$. Im folgenden nehmen wir $c = 1$ an.

Wir machen plausibel, dass die Erfolgschance für den (p-1)-Angriff bei zufälliger Wahl einer der von uns in Abschnitt 4.3 vorgeschlagenen Klassengruppen gering bzw. die Anwendung des (p-1)-Algorithmus aufwändiger ist als die Anwendung des schnellsten Index-Calculus-Algorithmus. In Abschnitt 4.1.6 werden wir sehen: Das Lösen des Wurzelproblems bzw. die Berechnung eines diskreten Logarithmus mit Buchmanns Algorithmus ist in Klassengruppen kubischer Stender-Körper K mit Diskriminante $\Delta \sim -2^{404}$ etwa genauso aufwändig wie das Faktorisieren einer RSA-1024-Bit-Zahl mit dem schnellsten heute bekannten Algorithmus, dem General Number Field Sieve.

Im Folgenden nehmen wir an, als obere Schranke B der Gruppenordnung sei $B = |\Delta|^{0.45}$ bekannt. (In kubischen Stender-Körpern ist B tatsächlich in der Größenordnung der Klassenzahl, siehe Abschnitt 4.4.2). Die Glattheitsschranke s setzen wir an als $s = B^{\frac{1}{u}}$ mit einer positiven reellen Zahl u. Sei A_1 die Anzahl der Gruppenoperationen, die ein Angreifer bereit sei zu investieren beim Versuch, einen konkreten Zahlkörper zu knacken, um also letztendlich ein konkretes Wurzelproblem oder diskretes Logarithmusproblem zu lösen. Pessimistisch nehmen wir an, ein Angreifer müsse nur erfolgreich den ersten Schritt unseres (p-1)-Algorithmus ausführen. Dann ergibt sich

$$A_1 = s \cdot \frac{\log B}{\log s} = 2^{\log_2 u + \log_2 |\Delta|^{\frac{0.45}{u}}} \quad .$$

Fassen wir A_1 als Funktion in der positiven rellen Veränderlichen u auf, so ist A_1 bei fixierter Diskriminante Δ streng monton wachsend.

Mit A_{Erfolg} bezeichnen wir die *erwartete* Anzahl der Gruppenoperationen, die ein Angreifer insgesamt aufwenden muss, um tatsächlich mit unserem (p-1)-Algorithmus im oben beschriebenem Sinne *einen* Zahlkörper zu brechen. Es gilt

$$A_{Erfolg} = \frac{A_1}{\Pr\left[\ h(\Delta) \text{ ist } s\text{-glatt } \right]} \quad .$$

Erste beispielhafte Annahme: $A_1 = 2^{28}$. A_1 entspricht damit etwa dem $\frac{1}{2^7} = \frac{1}{128}$ des Aufwandes zum Faktorisieren einer RSA-512-Bit-Zahl mit dem General Number Field Sieve. Durch Intervallschachtelung ergibt sich $u \approx 7.5$. Die angenommene Glattheitsschranke s ist $s = |\Delta|^{\frac{0.45}{u}} = 2^{404 \cdot \frac{0.45}{7.5}} \approx 2^{25}$. Die Zusammenhänge in den Abschnitten 4.4.2.1 und 4.4.2.2 ergeben $\Pr\left[\ h(\Delta) \text{ ist } s\text{-glatt } \right] \approx 2^{-22}$. Wir erhalten

$$A_{Erfolg} = \frac{A_1}{\Pr\left[\ h(\Delta) \text{ ist } s\text{-glatt } \right]} = 2^{50} \text{ Gruppenoperationen.}$$

Der erwartetete Gesamtaufwand für einen Angreifer entspricht somit etwa dem $\frac{1}{2^7} = \frac{1}{128}$ des Aufwandes zum Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve. Ein derartiger Angriff kann somit *in der Praxis unmöglich* erfolgreich durchgeführt werden.

Wir bemerken, dass sich dem Angreifer in einer kryptographischen Anwendung mit kleineren bis mittleren Benutzeranzahlen ein anderes Problem stellt: Mit großer Wahrscheinlichkeit verwendet überhaupt kein Benutzer einen Zahlkörper mit s-glatter Klassenzahl, so dass ein Angriff mit der (p-1)-Methode überhaupt keine Erfolgschance hat.

Nehmen beispielsweise $2^{10} = 1024$ Benutzer an der kryptographischen Anwendung teil, so ist

$$\begin{aligned}\Pr[\text{„Mind. 1 Benutzer hat } s\text{-glattes } h(\Delta)\text{“}] \\ = 1 - \Pr[\text{„Keiner hat } s\text{-glattes } h(\Delta)\text{“}] \\ = 1 - (1 - 2^{-22})^{1024} \approx 0.0002 \ .\end{aligned}$$

Zweite beispielhafte Annahme: $A_1 = 2^{40}$. A_1 entspricht damit etwa dem $2^5 = 32$-fachen des Aufwandes zum Faktorisieren einer RSA-512-Bit-Zahl mit dem General Number Field Sieve. Durch Intervallschachtelung ergibt sich $u \approx 5$. Die angenommene Glattheitsschranke s ist $s \approx 2^{37}$. Die Zusammenhänge in den Abschnitten 4.4.2.1 und 4.4.2.2 ergeben $\Pr[\ h(\Delta) \text{ ist } s\text{-glatt }] \approx 2^{-10}$. Wir erhalten

$$A_{Erfolg} = \frac{A_1}{\Pr[\ h(\Delta) \text{ ist } s\text{-glatt }]} = 2^{50} \text{ Gruppenoperationen.}$$

Der erwartete Gesamtaufwand für einen Angreifer entspricht somit auch hier etwa dem $\frac{1}{2^7} = \frac{1}{128}$ des Aufwandes zum Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve. Ein derartiger Angriff kann somit *in der Praxis unmöglich* erfolgreich durchgeführt werden.

Schlussfolgerung Zur Verhinderung des Erfolges unseres (p-1)-Algorithmus muss die Gruppenordnung einen hinreichend großen Primteiler enthalten. Bei Verwendung geeigneter Klassengruppen algebraischer Zahlkörper erscheint das Anwenden anderer Angriffe als des (p-1)-Algorithmus deutlich aussichtsreicher. Das Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve ist so aufwändig wie das Durchführen von 2^{57} Gruppenoperationen in der Klassengruppe eines kubischen Stender-Zahlkörpers (siehe oben). Unser (p-1)-Algorithmus benötigt hier mindestens 2^{57} viele Gruppenoperationen, wenn die Klassenzahl (Gruppenordnung) einen Primteiler $p \geqslant 2^{57}$ enthält.

4.1.6 Buchmanns Klassengruppen-Algorithmus

Buchmanns Klassengruppen-Algorithmus ([Buc89]) gehört zur Familie der Index-Calculus-Methoden. Buchmanns Algorithmus ist eine Verallgemeinerung des Algorithmus von Hafner und McCurley ([HM89]) zur Berechnung der Struktur von Klassengruppen. Während der Algorithmus von Hafner und McCurley lediglich auf imaginär-quadratische Zahlkörper angewendet werden kann, funktioniert Buchmanns Algorithmus in algebraischen Zahlkörpern K beliebigen Grades. Wenn die Struktur der Klassengruppe mit Diskriminante Δ bekannt ist, können die Klassenzahl $h(\Delta)$ und beliebige diskrete Logarithmen in der Klassengruppe effizient berechnet werden.

Seien $\alpha, \beta \in G = Cl(K)$ gegeben. Buchmanns Algorithmus berechnet Erzeuger $\gamma_1, \ldots, \gamma_m$ der zyklischen Untergruppen in G und deren Ordnungen $|\langle\gamma_1\rangle|, \ldots, |\langle\gamma_m\rangle|$: $G = \langle\gamma_1\rangle \ldots \langle\gamma_m\rangle$. Buchmanns Algorithmus gibt Informationen aus, die zur effizienten Berechnung der Darstellungen $\alpha = \prod_{i=1}^{m} \gamma_i^{a_i}, \beta = \prod_{i=1}^{m} \gamma_i^{b_i}$ mit ganzen Zahlen a_i, b_i genutzt werden können. Daher kann das Lösen des diskreten Logarithmusproblems

$$\alpha^x = \beta$$

darauf reduziert werden, $\prod_{i=1}^{m} \gamma_i^{a_i x} = \prod_{i=1}^{m} \gamma_i^{b_i}$ zu lösen, d.h. eine ganze Zahl x zu bestimmen, so dass gilt:

$$a_i x \equiv b_i \bmod |\langle \gamma_i \rangle|,\ i = 1, \ldots, m\ .$$

Dieses System simultaner Kongruenzen kann effizient mit dem Chinesischen Restsatz (siehe z.B. [Coh95]) gelöst werden.

Die Resultate aus Unterabschnitt 3.1.7 zeigen, dass Buchmanns Algorithmus alle in Kapitel 3 vorgestellten Kryptosysteme über algebraischen Zahlkörpern (NF-Kryptosysteme) bricht. Darüber hinaus wird sich zeigen: Buchmanns Algorithmus ist der schnellste heute bekannte Algorithmus, um die NF-Kryptosysteme zu brechen.

4.1.6.1 Der Algorithmus

Wir skizzieren nun Buchmanns Klassengruppen-Algorithmus. Eine ausführliche Darstellung findet sich in [Buc89]. Sei K ein algebraischer Zahlkörper vom Grad n mit Diskriminante Δ. Seine Klassengruppe bezeichnen wir mit $Cl(K)$. Bestimme zunächst für die Klassenzahl $h(\Delta)$ eine Schranke $h^*(\Delta)$, so dass $h^*(\Delta) < h(\Delta) < 2h^*(\Delta)$. Eine solche Schranke kann leicht berechnet werden (siehe [Coh95]). Nach Wahl eines Erzeugendensystems[2] $\mathrm{FB} = \{\mathfrak{x}_1, \ldots, \mathfrak{x}_m\}$, welches aus m Primidealen (die Anzahl m ist durch Bachs Konstante bestimmt, siehe [Bac90]) besteht, müssen zumindest m linear-unabhängige *Relationen* $(k_1, \ldots, k_m)$ gefunden werden, so dass $\prod_{i=1}^{m} [\mathfrak{x}_i]^{k_i} = [1]$.

Hierzu werden Ideale $\mathfrak{y} = \prod_{i=1}^{m} \mathfrak{x}_i^{e_i}$ berechnet, wobei $e_i \in \{0, 1, \ldots, 2h^*(\Delta)\}$ zufällig gewählt werden. Diese Ideale $\mathfrak{y}$ werden LLL-reduziert, was zu Idealen $\mathfrak{y}^*$ führt mit $[\mathfrak{y}^*] = [\mathfrak{y}]$.

Die LLL-reduzierten Ideale sollten jetzt in Faktoren aus der Menge FB faktorisiert werden, wenn möglich: $\mathfrak{y}^* = \prod_{i=1}^{m} \mathfrak{x}_i^{f_i}$. Dies führt zu einer Relation $(e_1 - f_1, \ldots, e_m - f_m)$, denn es gilt:

$$\prod_{i=1}^{m} [\mathfrak{x}_i]^{e_i - f_i} = [\mathfrak{y}][\mathfrak{y}]^{-1} = [1]. \tag{4.3}$$

Alle mittels (4.3) gefundenen Relationen werden als Spaltenvektor in die *Relationenmatrix* eingetragen. Die Abbildung

$$\begin{aligned} \phi : \mathbb{Z}^m &\rightarrow Cl(\Delta) \\ (e_1, \ldots, e_m) &\mapsto \prod_{i=1}^{m} [\mathfrak{x}_i]^{e_i} \end{aligned}$$

definiert einen Gruppen-Epimorphismus (d.h. einen surjektiven Homomorphismus von Gruppen). Daher ist der Kern $\Lambda = ker\,\phi$ von ϕ ein Teilgitter von $\mathbb{Z}^m$, und wir haben einen Gruppenisomorphismus $\mathbb{Z}^m/\Lambda \simeq Cl(K)$. Die Determinante $d = |\det(\Lambda)|$ der *vollen* Relationenmatrix (d.h. die Spaltenvektoren der Relationenmatrix erzeugen den vollständigen Kern Λ) ist daher gleich der Klassenzahl $h(\Delta)$. Im Algorithmus wird nach Auffinden einer neuen Relation und entsprechendem Erweitern der Relationenmatrix ständig d als Determinante der aktuellen Relationenmatrix berechnet. Wir erhalten $d = h(\Delta)$, wenn

$$h^*(\Delta) < d < 2h^*(\Delta)\ . \tag{4.4}$$

[2] Genauer: Nicht die Primideale erzeugen die Klassengruppe, sondern die Primidealklassen: $Cl(K) = \langle [\mathfrak{x}_1], \ldots, [\mathfrak{x}_m] \rangle$.

Falls d nicht innerhalb dieser Schranken liegt, aber größer als 0 ist, ist das von den Spaltenvektoren der Relationenmatrix erzeugte Gitter ein Teilgitter des Kerns Λ. Daher muss d in diesem Fall ein Vielfaches von $h(\Delta)$ sein. Zum Brechen einiger NF-Kryptosysteme wie RDSA genügt die Kenntnis eines Vielfachen der Klassenzahl. Wenn wir am Brechen derartiger NF-Kryptosysteme interessiert sind, berechnen wir aus Effizienzgründen nicht das volle Gitter Λ, sondern brechen Buchmanns Algorithmus an dieser Stelle ab. Ansonsten fahren wir mit der Relationengenerierung fort, bis (4.4) erfüllt ist und wir somit die Klassenzahl gefunden haben.

Die Berechnung der *Smith-Normalform (SNF)* (siehe z.B. [Coh95]) der jetzt erhaltenen Relationenmatrix sowie die Berechnung der zur SNF transformierenden unimodularen Matrizen führt zu Erzeugern $\gamma_1, \ldots, \gamma_m$ der zyklischen Untergruppen in G sowie zu deren Ordnungen $|\langle\gamma_1\rangle|, \ldots, |\langle\gamma_m\rangle|$: $G = \langle\gamma_1\rangle \ldots \langle\gamma_m\rangle$. Zudem kann man mit diesen Informationen zu vorgegebenen Gruppenelementen (Idealklassen) α, β effizient Darstellungen $\alpha = \prod_{i=1}^{m} \gamma_i^{a_i}, \beta = \prod_{i=1}^{m} \gamma_i^{b_i}$ finden mit ganzen Zahlen a_i, b_i.

4.1.6.2 Komplexität

Um die Laufzeit eines Algorithmus auszudrücken, die subexponentiell in der binären Länge einer Eingabegröße N ist, benutzen wir die folgende Funktion ($0 \leqslant u \leqslant 1$):

$$L(N)[u, v + o(1)] = e^{(v+o(1)) \ln(N)^u \ln(\ln(N))^{1-u}}$$

Beispielsweise ist $L(N)[0, v + o(1)] = \ln(N)^{v+o(1)}$ die Laufzeit eines in der Eingabelänge polynomiellen Algorithmus; $L(N)[1, v + o(1)] = e^{\ln(N)(v+o(1))}$ ist die Laufzeit eines in der Eingabelänge exponentiellen Algorithmus. Hingegen ist $L(N)[u, v + o(1)]$ mit $0 < u < 1$ die Laufzeit eines in der Eingabelänge sogenannten *subexponentiellen Algorithmus*.

Für quadratische Zahlkörper lässt sich beweisen (siehe [Buc89, S. 38]): Für $\beta \in \mathbb{R}_{>0}$ ist die Anzahl der $L(|\Delta|)[\frac{1}{2}, \beta + o(1)]$-glatten reduzierten Ideale $\sqrt{|\Delta|} \cdot L(|\Delta|)[\frac{1}{2}, -\frac{1}{4\beta} + o(1)]$. Buchmann nimmt in [Buc89] an, diese Aussage gelte auch für algebraische Zahlkörper höherer Grade. Wir bezeichnen diese Annahme als *Buchmanns Glattheitsannahme*. Unter der Verallgemeinerten Riemannschen Vermutung und Buchmanns Glattheitsannahme zeigt Buchmann [Buc89], dass man die Klassenzahl, die Struktur der Klassengruppe und den Regulator eines algebraischen Zahlkörpers von beliebigem, aber festem Grad n und Diskriminante Δ in Zeit $L(|\Delta|)[\frac{1}{2}, 1.7 + o(1)]$ berechnen kann. Bislang ging man davon aus, dass dieselbe Laufzeit zur Berechnung der Klassenzahl und der Struktur der Klassengruppe notwendig sei ([BBHM02], [MNP01]). Wir beweisen nun ein besseres Resultat:

Satz 4.1.1
Unter der Verallgemeinerten Riemannschen Vermutung und Buchmanns Glattheitsannahme benötigt Buchmanns Algorithmus zur Berechnung der Klassenzahl und der Struktur der Klassengruppe eines algebraischen Zahlkörpers von beliebigem, aber festem Grad n und Diskriminante Δ die erwartete Laufzeit

$$L(|\Delta|)\left[\frac{1}{2}, \sqrt{2} + o(1)\right].$$

Beweis: Gegeben sei ein algebraischer Zahlkörper K vom Grad n mit Diskriminante Δ. Für ein noch zu wählendes $\beta \in \mathbb{R}_{>0}$ setzen wir $c := L(|\Delta|)[\frac{1}{2}, \beta]$ und wählen in Buchmanns Algorithmus $\mathrm{FB} = \{\mathfrak{x} \mid \mathfrak{x} \text{ Primideal von } K,\ N(\mathfrak{x}) \leqslant c\}$. Die bzgl. der Laufzeit dominierenden Schritte von

Buchmanns Algorithmus haben gemäß [Buc89, S. 38f.] folgende erwartete Laufzeiten, wenn man die Gültigkeit der Verallgemeinerter Riemannschen Vermutung und von Buchmanns Glattheitsannahme voraussetzt:

a) Erzeugendensystem-Berechnung für das Relationengitter $L(|\Delta|)[\frac{1}{2}, 2\beta + \frac{1}{4\beta} + o(1)]$
b) Smith-Normalform-Berechnung für die Relationenmatrix $L(|\Delta|)[\frac{1}{2}, 4\beta + o(1)]$

Aus der Smith-Normalform lassen sich unmittelbar die Klassenzahl und die Struktur der Klassengruppe ablesen. Daher ergibt sich die erwartete Gesamtlaufzeit des Algorithmus zu

$$L(|\Delta|)[\frac{1}{2}, m(\beta) + o(1)], \qquad \text{wobei} \qquad m(\beta) \overset{\mathbb{R}_{\geq 0}}{=} \max\{2\beta + \frac{1}{4\beta}, 4\beta\}\,.$$

Wir bestimmen nun die positive reelle Zahl β_0, an der die Funktion $m(\beta)$ ihr Minimum annimmt. Für $a, b \in \mathbb{R}$ gilt $\frac{1}{2}\left(|a+b| + |a-b|\right) = \left\{ \begin{array}{l} a, \text{ falls } a \geqslant b \\ b, \text{ falls } a < b \end{array} \right\} = \max\{a, b\}$. Daher ist $m(\beta) = \frac{1}{2}\left(6\beta + \frac{1}{4\beta} + (-2\beta + \frac{1}{4\beta}\right) = 2\beta + \frac{1}{4\beta}$. Die Funktion m ist differenzierbar mit 1. Ableitung $m'(\beta) \overset{\mathbb{R}_{>0}}{=} 2 - \frac{1}{4\beta^2} = \frac{8\beta^2 - 1}{4\beta^2}$. Die 1. Ableitungsfunktion m' hat auf $\mathbb{R}_{>0}$ die Nullstelle $\beta_0 = \frac{1}{\sqrt{8}} = \frac{\sqrt{2}}{4}$ mit Vorzeichenwechsel von $-$ nach $+$ bei β_0. Da die Funktion m an den Rändern des Definitionsbereichs gegen ∞ geht, besitzt m den absoluten Tiefpunkt $T\left(\frac{\sqrt{2}}{4} \mid \sqrt{2}\right)$. Für die optimale Wahl von $\beta = \frac{\sqrt{2}}{4}$ erhalten wir die erwartete Laufzeit

$$L(|\Delta|)\left[\frac{1}{2}, m\left(\frac{\sqrt{2}}{4}\right) + o(1)\right] = L(|\Delta|)\left[\frac{1}{2}, \sqrt{2} + o(1)\right]\,.$$

□

Aus Satz 4.1.1 folgt: Buchmanns Algorithmus ist für betragsmäßig große Diskriminanten sehr ineffizient. Folglich sind die NF-Kryptosysteme aus Kapitel 3 für betragsmäßig große Diskriminanten nach heutigem Kenntnisstand sicher.

Zum Vergleich vergegenwärtigen wir uns, dass der effizienteste Algorithmus zum Faktorisieren ganzer Zahlen n (und damit insbesondere ein Algorithmus zum Brechen zahlreicher in der Praxis verwendeter Public-Key-Kryptosysteme wie z.B. RSA) eine viel kürzere Laufzeit von

$$L(n)\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}} + o(1)\right]$$

besitzt ([LV01]). Auch das Lösen der in Unterabschnitt 3.1 beschriebenen Berechnungsprobleme über Klassengruppen imaginär-quadratischer Zahlkörper der Diskriminante Δ verläuft effizienter: Empirische Untersuchungen legen hier die Laufzeit

$$L(|\Delta|)\left[\frac{1}{2}, 1 + o(1)\right]$$

viele Bitoperationen nahe. Vollmer [Vol00] beweist unter Annahme der Verallgemeinerten Riemann-Vermutung als obere Laufzeitschranke in imaginär-quadratischen Zahlkörpern

$$L(|\Delta|)\left[\frac{1}{2}, \frac{3}{4}\sqrt{2} + o(1)\right]\,,$$

wobei $\frac{3}{4}\sqrt{2} \approx 1.04$ ist.

4.1.6.3 Implementierung von Buchmanns Algorithmus

Zur experimentellen Untersuchung der tatsächlichen Laufzeit von Buchmanns Klassengruppenalgorithmus verwendeten wir die Implementierung aus dem Computeralgebra-System Pari/GP [BBB+02]. Wir optimierten den Algorithmus in der in Unterabschnitt 4.1.6.1 beschriebenen Weise, da wir – pessimistisch aus der Sicht der Benutzer der NF-Kryptosysteme – davon ausgehen, dass dem Angreifer zum Brechen der Kryptosysteme die Kenntnis eines Vielfachen der Klassenzahl genügt, was tatsächlich nicht für alle NF-Kryptosysteme gilt (siehe Kapitel 3).

Die resultierenden Laufzeiten, die wir in Anhang B auflisten, wurden auf Sun-Solaris-Workstations mit UltraSparc II-Prozessoren und 450MHz Taktfrequenz gemessen. In der Arbeit [HM00] wurde per Benchmark ermittelt, dass eine Sun-Solaris-Workstation mit UltraSparc I-Prozessor mit 170MHz Taktfrequenz 100 MIPS schnell ist (d.h. 100 Millionen Operationen pro Sekunde ausführt). Daher erscheint es vernünftig – erneut pessimistisch aus der Sicht der Benutzer der NF-Kryptosysteme – anzunehmen, dass unsere UltraSparc II-Prozessoren 400 MIPS schnell sind, d.h. sie führen 400 Millionen Operationen pro Sekunde aus. Wir haben bei unseren Experimenten als Stackgröße 900 MB realen physischen Speicher verwendet.

4.1.6.4 Schlussfolgerung

Zur Verhinderung des Erfolges des Buchmann-Algorithmus muss bei Verwendung von Klassengruppen algebraischer Zahlkörper die Diskriminante dem Betrage nach hinreichend groß gewählt werden. Insgesamt stellt sich heraus, dass bei Verwendung kryptographisch geeigneter Klassengruppen algebraischer Zahlkörper die Anwendung von Buchmanns Algorithmus im Allgemeinen für einen Angreifer am Aussichtsreichsten erscheint. Das Faktorisieren einer RSA-1024-Bit-Zahl mit dem General Number Field Sieve ist so aufwändig wie das Durchführen von 2^{57} Gruppenoperationen in der Klassengruppe eines kubischen Stender-Zahlkörpers (siehe oben). In Abschnitt 4.5 werden wir sehen: Buchmanns Algorithmus benötigt hier mindestens 2^{57} viele Gruppenoperationen, wenn $|\Delta| \geqslant 2^{404}$ gilt.

4.2 Notwendige Bedingungen für die kryptographische Eignung der Gruppe

Um das gewünschte Sicherheitsniveau der NF-Kryptoverfahren zu gewährleisten, müssen wir unsere Gruppe so wählen, dass keiner der in Abschnitt 4.1 vorgestellten Angriffe in der Praxis Erfolg hat. Das verhindern wir dadurch, dass wir die Parameter groß machen, die die Komplexität der jeweiligen Angriffe explodieren lässt. Informal betrachten wir ein kryptographisches Verfahren in diesem Abschnitt als sicher, wenn zu dessen Brechen mindestens $6.01 \cdot 10^{10}$ MIPS-Jahre aufgebracht werden müssten. Dies entspricht dem Aufwand zur Faktorisierung einer RSA-1024-Bit-Zahl mit dem schnellsten hierfür bekannten Algorithmus, dem General Number Field Sieve.

Es ergeben sich die notwendigen (und nach heutigem Kenntnisstand hinreichenden) Bedingungen für eine kryptographisch geeignete Klassengruppe $Cl(K)$ eines algebraischen Zahlkörpers K vom Grade > 2:

$R(K)$ klein. Der Regulator $R(K)$ muss hinreichend klein sein.

$|\Delta|$ groß. Die Diskriminante $|\Delta|$ muss dem Betrage nach hinreichend groß gewählt werden. Zum Erreichen des Sicherheitsniveaus des RSA-Verfahrens mit 1024 Bit Schlüssellänge muss für kubische Zahlkörper beispielsweise $|\Delta| \geqslant 2^{404}$ gewählt werden.

Große(r) Primteiler. Die Gruppenordnung (Klassenzahl) muss hinreichend große Primteiler enthalten. Zur Präzisierung dieser Forderung sei $|G| = |Cl(K)| = \prod_{i=1}^{k} p_i^{e(p_i)}$ die Primfaktorzerlegung von $|G|$, $p_i \neq p_j$ für $i \neq j$. Dann muss eine Teilmenge $S \subseteq \{p_1, \ldots, p_k\}$ mit den folgenden drei Eigenschaften existieren:

$$\prod_{p \in S} p \geqslant 2^{114} \tag{4.5}$$

$$\text{Es existiert } p \in S \text{ mit} \quad p \geqslant 2^{57} \tag{4.6}$$

$$\text{Es existiert } 0 < \varepsilon \ll 1 \text{ mit} \quad \sum_{p \in S} \frac{1}{p^{e(p)}} \leqslant \varepsilon \tag{4.7}$$

Als sehr kleine positive Zahl ε im Sinne von (4.7) betrachten wir z.B. $\varepsilon = 2^{-24}$.
Beispiel 1: Die Bedingung „große(r) Primteiler“ ist erfüllt, wenn ein sehr großer Primteiler von $|G|$ existiert: $S = \{p_1\}$, $p_1 \geqslant 2^{114}$.
Beispiel 2: Die Bedingung „große(r) Primteiler“ ist erfüllt, wenn zwei Primteiler von $|G|$ mittlerer Größe existieren: $S = \{p_1, p_2\}$, $p_1 p_2 \geqslant 2^{114}$, $\frac{1}{p_1^{e(p_1)}} + \frac{1}{p_2^{e(p_2)}} \leqslant 2^{-24}$, also z.B. $p_1, p_2 \geqslant 2^{57}$.

Bemerkung: Großes $h(\Delta)$ als implizite Forderung. Die zweite und dritte Bedingung implizieren bereits eine große Klassenzahl $h(\Delta) = h(K)$. Denn aus $|\Delta| \geqslant 2^{404}$ folgt aus dem Satz von Breuer-Siegel (siehe z.B. [Coh95]) bereits $h(K)R(K) \sim \sqrt{|\Delta|}$ und wegen der ersten Forderung folgt hieraus, dass die Klassenzahl $h(K)$ groß ist.

Wir erläutern die Notwendigkeit obiger Bedingungen und zeigen auf, wie wir die Gültigkeit der Bedingung bei der Auswahl der Klassengruppe sicher stellen können:

1. $R(K)$ ***klein.*** Der Regulator $R(K)$ muss hinreichend klein sein, damit die Gleichheit zweier gegebener Elemente in $Cl(K)$ effizient entschieden werden kann. Diese Bedingung ist somit für die Effizienz des NF-Kryptoverfahrens unverzichtbar. Hinsichtlich der Sicherheit des NF-Kryptoverfahrens implizieren die Forderungen „$R(K)$ klein“ und „$|\Delta|$ groß“ eine große Klassenzahl $h(K)$, wie wir oben bereits begründet haben. Zur Gewährleistung eines gewünschten Sicherheitslevels benötigen wir eine große Klassenzahl. Daher ist die Forderung „$R(K)$ klein“ auch indirekt für die Sicherheit der NF-Kryptoverfahren relevant.

2. $|\Delta|$ ***groß.*** Diese Bedingung verhindert den Erfolg von Index-Calculus-Methoden, insbesondere den Erfolg von Buchmanns Klassengruppen-Algorithmus. Die Größe des Absolutbetrages $|\Delta|$ der Diskriminante einer prinzipiell kryptographisch geeigneten Zahlkörperfamilie steuert direkt den Aufwand, der nach heutigem Kenntnisstand zum Brechen der NF-Kryptoverfahren betrieben werden muss. Beispielsweise muss für kubische Zahlkörper $|\Delta| \geqslant 2^{404}$ gewählt werden, um das Sicherheitsniveau des RSA-Verfahrens mit 1024 Bit Schlüssellänge zu erreichen. Die Beziehung zwischen der Größe von $|\Delta|$ und dem erreichten Sicherheitslevel arbeiten wir detailliert in Abschnitt 4.5 aus.

3. ***Große(r) Primteiler.*** Diese Bedingung verhindert den Erfolg der folgenden Algorithmen:

- Vollständiges Durchprobieren. Denn der erwartete Aufwand wären $\frac{1}{2}\operatorname{ord}\alpha \geqslant 2^{113} \gg 2^{57}$ Gruppenoperationen, weil gemäß Korollar 4.2.2 mit äußerst großer Wahrscheinlichkeit $\operatorname{ord}\alpha \geqslant 2^{114}$ gilt.
- Baby-Step-Giant-Step-Algorithmus. Denn der erwartete Mindestaufwand wären $\sqrt{\operatorname{ord}\alpha} \geqslant 2^{57}$ Gruppenoperationen, weil gemäß Korollar 4.2.2 mit äußerst großer Wahrscheinlichkeit $\operatorname{ord}\alpha \geqslant 2^{114}$ gilt.
- Pollards Lambda-Algorithmus. Denn der erwartete Mindestaufwand wären $\sqrt{\operatorname{ord}\alpha} \geqslant 2^{57}$ Gruppenoperationen, weil gemäß Korollar 4.2.2 mit äußerst großer Wahrscheinlichkeit $\operatorname{ord}\alpha \geqslant 2^{114}$ gilt.
- Unser (p-1)-Algorithmus. Denn der erwartete Mindestaufwand wären 2^{57} Gruppenoperationen, weil $\operatorname{ord}\alpha$ nach Thoerem 4.2.1 mit äußerst großer Wahrscheinlichkeit einen Primteiler $p \geqslant 2^{57}$ besitzt.

4. ***Kombination aller drei Bedingungen.*** Die Kombination aller drei genannten Bedingungen verhindert die Berechnung eines Vielfachen der Elementordnung eines Gruppenelementes und somit den Erfolg des Pohlig-Hellman-Algorithmus.

Satz 4.2.1
Sei G eine endliche Abelsche Gruppe, und sei $|G| = \prod_{i=1}^{k} p_i^{e(p_i)}$ die Primfaktorzerlegung von $|G|$, $p_i \neq p_j$ für $i \neq j$. Sei $\alpha \in G$ zufällig mit Gleichverteilung gewählt. Es sei $S \subseteq \{p_1, \ldots, p_k\}$. Dann gilt:

$$\Pr\left[\prod_{p\in S} p \mid \operatorname{ord}\alpha\right] \geqslant 1 - \sum_{p\in S} \frac{1}{p^{e(p)}}$$

Beweis: Sei $F := \{p_1, \ldots, p_k\}$ die Menge der Primteiler von $|G|$. Als endliche Abelsche Gruppe ist G das innere direkte Produkt seiner p-Sylow-Gruppen S_p:

$$G = \prod_{p\in F} S_p \ ,$$

wobei die Menge S_p für jedes $p \in F$ aus allen Gruppenelementen mit Primzahlpotenz p^i als Elementordnung besteht (für ein $i \in \{0, \ldots, e(p)\}$) und jedes $\alpha \in G$ eine eindeutige Darstellung $\alpha = \prod_{p\in F} \alpha_p$ mit $\alpha_p \in S_p$ besitzt. Es gilt $\operatorname{ord}\alpha = kgV\{\operatorname{ord}\alpha_p : \ p \in F\}$. Wir können ein Gruppenelement $\alpha \in G$ zufällig mit Gleichverteilung wählen, indem wir für alle $p \in F$ zufällige Elemente $\alpha_p \in S_p$ mit Gleichverteilung auswählen und das Produkt $\alpha := \prod_{p\in F} \alpha_p$ bilden. Daher erhalten wir

$$\begin{aligned}
\Pr\left[\prod_{p\in S} p \mid \operatorname{ord}\alpha\right] &= \Pr[p \mid \operatorname{ord}\alpha \text{ für alle } p \in S] \\
&= 1 - \Pr[p \nmid \operatorname{ord}\alpha \text{ für mindestens ein } p \in S] \\
&= \begin{cases} 1 - \sum_{p\in S} \Pr[p \nmid \operatorname{ord}\alpha], & \text{falls } |S| = 1 \\ 1 - \left(\sum_{p\in S} \Pr[p \nmid \operatorname{ord}\alpha] - \Pr[p \nmid \operatorname{ord}\alpha \text{ für alle } p \in S]\right), & \text{falls } |S| > 1 \end{cases}
\end{aligned}$$

Für jedes $p \in S$ existiert genau ein Element in S_p, dessen Ordnung nicht teilbar durch p ist (nämlich das neutrale Element der Gruppe).

$$\Pr\left[\prod_{p\in S} p \mid \operatorname{ord}\alpha\right] = \begin{cases} 1 - \sum_{p\in S} \frac{1}{p^{e(p)}}, & \text{falls } |S| = 1 \\ 1 - \sum_{p\in S} \frac{1}{p^{e(p)}} + \prod_{p\in S} \frac{1}{p^{e(p)}}, & \text{falls } |S| > 1 \end{cases}$$

Dies zeigt die Behauptung. □

Korollar 4.2.2
Sei G eine endliche Abelsche Gruppe, und sei $|G| = \prod_{i=1}^{k} p_i^{e(p_i)}$ die Primfaktorzerlegung von $|G|$, $p_i \neq p_j$ für $i \neq j$. Sei $\alpha \in G$ zufällig mit Gleichverteilung gewählt. Es existiere $S \subseteq \{p_1, \ldots, p_k\}$ mit $\prod_{p \in S} p \geqslant 2^{114}$. Dann gilt:

$$\Pr\left[\operatorname{ord}\alpha \geqslant 2^{114}\right] \geqslant 1 - \sum_{p \in S} \frac{1}{p^{e(p)}}$$

Beweis: Aus Satz 4.2.1 folgt unmittelbar, dass $\operatorname{ord}\alpha$ mit mindestens der angegebenen Wahrscheinlichkeit paarweise verschiedene Primteiler besitzt, deren Produkt größer als 2^{114} ist. Hieraus folgt direkt die Behauptung. □

4.3 Erzeugen geeigneter Klassengruppen

Durch den *Satz von Brauer–Siegel* (*analytische Klassenzahlformel*, siehe Abschnitt 4.4.1) wissen wir: Für genügend großen Absolutbetrag $|\Delta|$ der Diskriminante ist das Produkt aus Regulator und Klassenzahl in der Größenordnung von $\sqrt{|\Delta|}$. Wie unsere Experimente und die von Neis [Nei02] zeigen, gilt bereits $hR \sim \sqrt{|\Delta|}$ für Diskriminanten einiger Hundert Bits. Jedoch sind wir mit dem Problem konfrontiert, dass der Regulator typischerweise groß und die Klassenzahl klein ist (siehe [CL83, CM87, CM90]). Hierdurch sind zufällig gewählte algebraische Zahlkörper mit großem Absolutbetrag der Diskriminante im Allgemeinen für die von uns vorgestellten NF-Kryptosysteme unbrauchbar; denn ein großer Regulator lässt keinen effizienten Gleichheitsentscheid in der Klassengruppe zu; zudem würde eine kleine Klasenzahl erfolgreiche Angriffe auf die NF-Kryptosysteme zulassen (vollständiges Durchprobieren oder (p-1)-Methode, siehe Abschnitt 4.1).

Wir werden in diesem Abschnitt aufzeigen, dass es unendliche Familien von Zahlkörpern mit kleinen Regulatoren und somit großen Klassenzahlen gibt. In den darauf folgenden Abschnitten werden wir sehen, dass diese Zahlkörperfamilien auch die ansonsten für die Effizienz und Sicherheit der NF-Kryptosysteme geforderten Eigenschaften erfüllen.

Wir betrachten kurz den Spezialfall quadratischer Zahlkörper. Imaginär-quadratische Zahlkörper haben immer den Regulator $R = 1$ und somit große Klassenzahlen h, sofern die Diskriminante dem Betrage nach groß gewählt wird. Wir zeigen in [BBHM99] (siehe auch [Ham02]), weshalb ihre Maximalordnungen für kryptographische Anwendungen geeignet sind. In [BBHM99] geben wir auch Beispiele für Maximalordnungen reell-quadratischer Zahlkörper, welche die Bedingungen aus Abschnitt 4.2 erfüllen.

In der vorliegenden Arbeit beschäftigen wir uns mit Zahlkörpern vom Grad > 2, weil wir ausnutzen wollen, dass die Komplexität der kryptographisch-algorithmischen Probleme mit wachsendem Körpergrad exponentiell wächst.

Im Folgenden stellen wir kryptographisch geeignete Familien von Ordnungen algebraischer Zahlkörper der Grade $3, 4, 6$ vor. Im letzten Unterabschnitt präsentieren wir auch Zahlkörper deutlich größerer Grade, für die man große Primteiler in der Klassenzahl explizit angeben kann.

4.3.1 Stender-Körper

Stender [Ste75] betrachtet algebraische Zahlkörper

$$K = \mathbb{Q}(\sqrt[n]{D^n \pm d}), \tag{4.8}$$

mit erzeugendem Polynom $f(x) = x^n - (D^n \pm d)$, wobei $n \in \{3, 4, 6\}$, $D, d \in \mathbb{Z}_{>0}$, und d eine weitere Bedingung erfüllt. Wir nennen diese Zahlkörper *Stender-Körper*. Stender bestimmt explizit ein System von Fundamentaleinheiten von K. Dies führt zum exakten Wert des Regulators. In Abschnitt 4.3 werden wir sehen, dass die Klassenzahlen der Stender-Körper (zumindest im Spezialfall $d = 1$) groß sind, während ihre Regulatoren klein sind. Für Stender-Körper weisen wir in Abschnitt 4.4.2 zudem unter gewissen plausiblen Annahmen theoretisch und experimentell nach, dass die Klassenzahl hinreichend große Primteiler enthält.

In Stender-Körpern vom Grad $n = 3$ lassen sich die NF-Kryptosysteme besonders effizient implementieren. Insbesondere kann die Gleichheit zweier gegebener Idealklassen sehr effizient entschieden werden; hier empfehlen wir den Parameter $d = 1$, was wir in Abschnitt 4.4.1 begründen werden.

4.3.2 Buchmanns Zahlkörper vom Grad 4

Sei $K = \mathbb{Q}(\sqrt[4]{-D})$, $D = 4k^4 + d$, $k \in \mathbb{Z}_{>0}$, $d \in \mathbb{Z}$ wobei $0 < |d| \leqslant 4k$. Wir betrachten die Ordnung $\mathcal{O} = \mathbb{Z}[\sqrt[4]{-D}]$ von K.

Wir empfehlen die Wahl des Parameters $d = 1$; denn dies ermöglicht den effizientesten Gleichheitstest, wie wir in Abschnitt 4.4.1 sehen werden.

4.3.3 Reelle kubische Zahlkörper

Sei ρ_ℓ die eindeutig bestimmte reelle Wurzel des irreduziblen Polynoms $f(X) = X^3 + \ell X - 1$ ($\ell \in \mathbb{Z}_{\geqslant 1}$), und sei $K = \mathbb{Q}(\rho_\ell)$. Dann ist $\epsilon_\ell := \frac{1}{\rho_\ell}$ eine Fundamentaleineit von K ([Lou95]), und es gilt $\ell < \epsilon_\ell < \ell + 1$.

Wir wählen ℓ so, dass $3^2 \nmid \ell$ und $p^2 \nmid \Delta_\ell = 4\ell^3 + 27$ für alle Primzahlen $p \geqslant 5$. Ein geeignetes ℓ liegt beispielsweise vor, wenn der Absolutbetrag Δ_ℓ der Diskriminante von K prim ist. Gemäß [Lou95] ist $\mathcal{O} = \mathbb{Z}[\epsilon_\ell]$ die Maximalordnung von K.

4.3.4 Total-imaginäre Zahlkörper vom Grad 4

Sei $K = \mathbb{Q}(i, \sqrt{\delta})$, wobei $\delta = m^2 + 4i$ quadratfrei in $\mathbb{Z}[i]$, $m = a + ib \in \mathbb{Z}[i]$ mit $a > b \geqslant 0$ und $a \not\equiv b \bmod 2$. Sei $D := |\delta|^2$.

Geeignete Instanzen dieser Körper finden wir beispielsweise, indem wir eine Primzahl $p > 2$ wählen mit $5 \nmid p^2 + 1$ und $a := 2p$, $b := p$, $m := a + ib$, $\delta := m^2 + 4i$ setzen.

4.3.5 p-te Kreiskörper

Seien p eine Primzahl und ζ_p eine primitive p-te Einheitswurzel (z.B. $\zeta_p = e^{2i\pi/p}$), dann heißt $K = \mathbb{Q}(\zeta_p)$ *p-ter Kreiskörper (pth cyclotomic field)*. Seine Diskriminante ist $\Delta \in \{\pm p^{p-2}\}$. Der Grad von K ist $\varphi(p) = p - 1$. Die Klassenzahl von K ist

$$h_p = h_p^- \cdot h_p^+ \ ,$$

wobei die *Relativklassenzahl* $h_p^- \in \mathbb{Z}_{>0}$ groß und effizient berechenbar ist und $h_p^+ \in \mathbb{Z}_{>0}$ klein und schwierig zu berechnen ist ([Was97], siehe auch Abschnitt 4.4.1).

Wir schlagen die Wahl von $p \in \{367, 373, 431, 443, 457, 461, 491\}$ vor; denn wir wissen, dass die entsprechenden Kreiskörper eine hinreichend große Klassenzahl und einen hinreichend großen Primteiler besitzen (siehe Abschnitt 4.4.1). Der Primteiler ist hier explizit bekannt genau wie ein sehr großer Faktor der Klassenzahl. Die Situation ist hier also durchaus vergleichbar mit der Situation der Punktegruppe einer kryptographisch geeigneten elliptischen Kurve.

Für $p > 491$ gibt es viele weitere Beispiele p-ter Kreiskörper mit guten kryptographischen Eigenschaften. Da aber der Grad dieser Zahlkörper $p - 1$ ist, werden entsprechende Implementierungen der NF-Krpytosysteme mit wachsendem Grad immer ineffizienter. Da die Klassenzahl (bzw. ein kleines Vielfaches) explizit bekannt ist, können hier auch die üblichen Signaturverfahren und sonstigen kryptographischen Protokolle angewendet werden, die die Kenntnis der Gruppenordnung voraussetzen.

Heute kann zwar niemand in Kreiskörpern vom Grad $p - 1 \geqslant 366$ so effizient rechnen, dass kryptographische Anwendungen hier für die Praxis interessant würden. Allerdings scheinen Optimierungen der Arithmetik in Kreiskörpern auf Grund deren reichen Struktur (siehe [Was97]) möglich. Diese sind Gegenstand künftiger Forschung.

4.4 Nachweis der kryptographischen Eignung

In diesem Abschnitt zeigen wir, dass die in Abschnitt 4.3 vorgeschlagenen Klassengruppen $Cl(K)$ algebraischer Zahlkörper K mit Diskriminante Δ, Regulator $R(K)$ und Klassenzahl $h(K)$ kryptographisch geeignet sind. Dazu haben wir alle in Abschnitt 4.2 aufgeführten Eigenschaften nachzuweisen:

$|\Delta|$ groß. Der Nachweis einer großen Diskriminante ist trivial, da wir die Diskriminante im konkreten Fall einfach ausrechnen können. Die Parameter des Zahlkörpers wählen wir jeweils so, dass $|\Delta|$ die gewünschte Größenordnung hat. Mit dieser Bedingung werden wir uns im Folgenden nicht weiter beschäftigen.

$R(K)$ klein. Wir hatten oben bereits erläutert, dass $R(K)h(K) \sim \sqrt{|\Delta|}$. Großer Absolutbetrag der Diskriminante und kleiner Regulator $R(K)$ implizieren somit große Klassenzahl $h(K)$. Wir weisen diese Forderung anhand theoretischer Analysen nach und untermauern unsere Ergebnisse durch experimentelle Untersuchungen (Abschnitt 4.4.1).

Große(r) Primteiler. In Abschnitt 4.4.2 weisen wir nach, dass die Klassenzahl jeweils mit sehr großer Wahrscheinlichkeit hinreichend große Primteiler besitzt. Zunächst führen wir unter gewissen Annahmen und unter Verwendung der Heuristik von Cohen-Martinet den theoretischen Nachweis (Unterabschnitt 4.4.2.1). Hierzu vergleichen wir die Wahrscheinlichkeit,

dass die Klassenzahl nur Primteiler unterhalb einer Schranke B hat, mit der entsprechenden Wahrscheinlichkeit für eine ganze Zufallszahl aus der Größenordnung der Klassenzahl. Wir führen die entsprechende Analyse beispielhaft an kubischen Stender-Körpern. Unsere experimentellen Untersuchungen aus Unterabschnitt 4.4.2.2 stärken unsere zuvor theoretisch gewonnenen Einsichten.

4.4.1 Sehr kleiner Regulator und sehr große Klassenzahl

In diesem Abschnitt zeigen wir, dass die von uns vorgeschlagenen Klassengruppen einen sehr kleinen Regulator besitzen. Mit wachsendem Absolutbetrag der Diskriminante wächst damit die Klassenzahl.

Die Berechnung der Klassenzahl $h(K)$ oder diskreter Logarithmen in einem algebraischen Zahlkörper K ist ein sehr schwieriges Berechnungsproblem. Die Komplexität dieses Berechnungsproblems ist subexponentiell in der binären Länge der Diskriminante und exponentiell im Grad des Zahlkörpers ([BP97]). Im Spezialfall imaginär-quadratischer Zahlkörper ist die Berechnung der Klassenzahl oder diskreter Logarithmen mindestens so schwierig wie das Faktorisieren ganzer Zahlen, die Produkt zweier großer Primzahlen sind ([Sch82], siehe auch Abschnitt 3.1.7).

Wir haben in Abschnitt 4.3 bereits erwähnt, dass die Diskriminante Δ, die Klassenzahl h und der Regulator R eines algebraischen Zahlkörpers K vom Grad n über die *analytische Klassenzahlformel (Satz von Brauer–Siegel)* in Beziehung stehen. Der exakte Zusammenhang lautet wie folgt (siehe [BS66]):

$$h = \frac{\kappa w \sqrt{|\Delta|}}{2^{r+s}\pi^s R} \ . \tag{4.9}$$

Hierbei ist κ der Wert der ζ-Funktion von K an der Stelle 1; w ist die Anzahl der Einheitswurzeln in K (typischerweise haben wir $w = 2$); r und s sind positive ganze Zahlen mit $r + 2s = n$; dabei ist r die Anzahl der reellen Einbettungen, $2s$ die Anzahl der komplexen Einbettungen von K in $\mathbb{C}$.

Imaginär-quadratische Zahlkörper Imaginär-quadratische Zahlkörper haben immer Regulator $R = 1$, somit hinreichend große Klassenzahl, sofern der Betrag der Diskriminante groß ist. Tatsächlich gilt $h \sim \sqrt{|\Delta|}$ für kryptographisch relevante Größenordnungen der Diskriminante. Nach [Cox89] besitzen die Klassenzahlen imaginär-quadratischer Zahlkörper folgende untere Schranke:

$$h > \frac{1}{7000} \prod_{p||\Delta|} \left(1 - \frac{\lfloor 2\sqrt{p} \rfloor}{p+1}\right) \ln |\Delta| \ . \tag{4.10}$$

Detaillierte Analysen zu imaginär-quadratischen Zahlkörpern finden sich in [Ham02].

4.4.1.1 Stender-Körper

Wir betrachten die Stender-Körper $K = \mathbb{Q}(\sqrt[n]{D^n \pm d})$ vom Grade $n \in \{3, 4, 6\}$, wobei $D \in \mathbb{Z}_{>16}$, so dass $D^n \pm 1$ quadratfrei ist; sei $d \in \{\pm 1\}$. Stender [Ste75] bestimmt explizit ein System von Fundamentaleinheiten von K. Somit kann explizit der Regulator von K berechnet werden. Im Anhang A der vorliegenden Arbeit bestimmen wir in geschlossener Form obere Schranken für die Regulatoren der Stender-Körper und leiten hieraus explizite untere Schranken für die Klassenzahlen her. Die entsprechenden Ergebnisse listen wir in Tabelle 4.1 auf. Für kubische Stender-Körper

Grad	Obere Schranke für Regulator	Untere Schranke für Klassenzahl
$n = 3$	$R \leqslant \frac{2}{3}\ln(3 \cdot (D^3 \pm 1))$	$h \geqslant \frac{1}{6}\sqrt{\frac{D^3 \pm 1}{\ln^3(3 \cdot (D^3 \pm 1))}}$
$n = 4$	$R < 4 \cdot (\ln(\sqrt{3} \cdot D))^2$	$h > \frac{c}{64\pi} \cdot \frac{D^3}{(\ln D)^2}$
$n = 6$	$R < 9324 \cdot \ln(2D)$	$h > \frac{c}{35812448 \cdot \pi^2} \cdot \frac{D^{10}}{\ln(2D)}$

Tabelle 4.1: Schranken für Regulator und Klassenzahl von Stender-Körpern, $d = 1, D > 16$

Fall	# reduzierter Hauptideale in $\mathcal{O}$
$d = 1$	1
$d > 1$	2
$k \geqslant 2$ und $d = -1$	2
$k \geqslant 2$ und $d < -1$ und $d \mid 4k$	4

Tabelle 4.2: Anzahl reduzierter Hauptideale in Buchmanns Zahlkörpern vom Grad 4

zitieren wir dabei das Ergebnis von Louboutin ([Lou95]), der mit einem anderen Ansatz explizite Schranken angeben kann. Die positive Konstante c aus Tabelle 4.1 lässt sich effektiv berechnen ([Sta74]). Man erkennt, dass die Klassenzahlen mit wachsendem Absolutbetrage der Diskriminante hinreichend schnell wachsen, während die Regulatoren verhältnismäßig klein bleiben. Unsere expliziten Berechnungen aus Abschnitt 4.4.2.2 stützen unsere theoretischen Resultate und zeigen, dass man in Tabelle 4.1 die Konstante $c \gg 10^9$ setzen kann.

Für den in Abschnitt 4.3 für kubische Stender-Körper empfohlenen Parameter $d = 1$ ist die Periodenlänge der Maximalordnung mit der Ausnahme sehr weniger, einfach zu prüfender Fälle exakt 1 (d.h. $\mathcal{O}$ ist das einzige reduzierte Hauptideal), wenn $D \not\equiv 0 \bmod 3$ gewählt wird ([Wil90]).

4.4.1.2 Buchmanns Zahlkörper vom Grad 4.

Sei $K = \mathbb{Q}(\sqrt[4]{-D})$, $D = 4k^4 + d$, $k \in \mathbb{Z}_{>0}$, $d \in \mathbb{Z}$ wobei $0 < |d| \leqslant 4k$. Wir betrachten die Ordnung $\mathcal{O} = \mathbb{Z}(\sqrt[4]{-D})$ von K.

In Abhängigkeit der Werte von d und k berechnete Buchmann [Buc87a] die Fundamentaleinheit von $\mathcal{O}$ sowie die Menge aller Minima von $\mathcal{O}$. Buchmanns Resultat ist aus folgenden Gründen wertvoll für uns: Erstens können wir mit der Kenntnis der Fundamentaleinheit von $\mathcal{O}$ nachweisen, dass der Regulator von $\mathcal{O}$ sehr klein, die Klassenzahl folglich sehr groß ist. Zweitens sind die Inversen der Minima gerade die Erzeuger der reduzierten Hauptideale; gemäß [Buc87a] ist die Anzahl der reduzierten Hauptideale in allen Fällen sehr klein, wie Tabelle 4.2 zeigt:

Der Regulator R ist proportional zur Anzahl der reduzierten Hauptideale; daher ist der Regulator hier hinreichend klein. Bei Benutzung der Ordnung $\mathcal{O}$ des vorgestellten Zahlkörpers können wir also einfach den Zyklus der reduzierten Hauptideale in $\mathcal{O}$ berechnen und somit effizient die

Gleichheit von Idealklassen entscheiden. Offensichtlich erlaubt die Wahl des Parameters $d = 1$ einen besonders effizienten Gleichheitstest für Idealklassen.

4.4.1.3 Reelle kubische Zahlkörper

Sei ρ_ℓ die eindeutig bestimmte reelle Wurzel des irreduziblen Polynoms $f(X) = X^3 + \ell X - 1$ ($\ell \in \mathbb{Z}_{\geqslant 1}$), und sei $K = \mathbb{Q}(\rho_\ell)$. Wenn ℓ wie in Abschnitt 4.3 angegeben gewählt wird, erhalten wir mit $\epsilon_\ell := \frac{1}{\rho_\ell}$ die Maximalordnung $\mathcal{O} = \mathbb{Z}[\epsilon_\ell]$, und nach [Lou95] erhalten wir für Diskriminanten $\Delta_\ell = 4\ell^3 + 27 \geqslant 2 \cdot 10^5$

$$h \geqslant \frac{\Delta_\ell}{20 \ln^2 \Delta_\ell} \ ,$$

was gewährleistet, dass die Klassenzahl hinreichend schnell mit wachsender Diskriminante wächst.

4.4.1.4 Total-imaginäre Zahlkörper vom Grad 4

Sei $K = \mathbb{Q}(i, \sqrt{\delta})$, wobei $\delta = m^2 + 4i$ quadratfrei in $\mathbb{Z}[i]$, $m = a + ib \in \mathbb{Z}[i]$ mit $a > b \geqslant 0$ und $a \not\equiv b \bmod 2$. Sei $D := |\delta|^2$. Aus [Lou94] erhalten wir für $D \geqslant 2 \cdot 10^9$:

$$R \leqslant \ln(\sqrt{D}) \text{ und } h \geqslant \frac{1}{17} \frac{\sqrt{D}}{\ln^2 D} \ .$$

Es gilt $|\Delta| = 16D$. Daher wächst die Klassenzahl hinreichend schnell mit wachsendem Parameter $|\delta|$.

4.4.1.5 p-te Kreiskörper

Seien p eine Primzahl und ζ_p eine primitive p-te Einheitswurzel, dann betrachten wir den p-ten Kreiskörper $K = \mathbb{Q}(\zeta_p)$. Dieser besitzt die Diskriminante $\Delta \in \{\pm p^{p-2}\}$ und den Grad $\varphi(p) = p-1$. Die Klassenzahl von K zerfällt in ein Produkt

$$h_p = h_p^- \cdot h_p^+ \ ,$$

wobei die Relativklassenzahl $h_p^- \in \mathbb{Z}_{>0}$ groß und effizient berechenbar ist und $h_p^+ \in \mathbb{Z}_{>0}$ klein (im Allgemeinen 1, in allen Beispielen der Tabelle $\leqslant 8$) und schwierig zu berechnen ist ([Was97]).

Für die Relativklassenzahl gilt für $p \to \infty$

$$\log h_p^- \sim \frac{1}{4} \varphi(p) \log p$$

Lehmer und Masley [LM78] berechneten die Relativklassenzahlen der p-ten Kreiskörper für die Primzahlen p mit $200 < p < 521$. Darüber hinaus führende Berechnungen finden sich in [Sho99]. Die entsprechenden Relativklassenzahlen und damit auch die Klassenzahlen haben gute kryptographische Eigenschaften. Die Tabelle 4.3 zeigt einige Beispiele kryptographisch geeigneter p-ter Kreiskörper. Sie besitzen eine große Relativklassenzahl, welche jeweils einen sehr großen Primteiler enthält. Sowohl Relativklassenzahl als auch großer Primteiler sind explizit bekannt und können in [LM78] nachgelesen werden. Aus Platzgründen drucken wir die Zahlen nicht explizit in Tabelle 4.3 ab, sondern geben nur deren Größenordnung an.

p	h_p^-	Größter Primteiler
367	$\sim 10^{91}$	$\sim 10^{57}$
373	$\sim 10^{93}$	$\sim 10^{54}$
431	$\sim 10^{114}$	$\sim 10^{78}$
443	$\sim 10^{119}$	$\sim 10^{93}$
457	$\sim 10^{124}$	$\sim 10^{75}$
461	$\sim 10^{125}$	$\sim 10^{92}$
491	$\sim 10^{137}$	$\sim 10^{68}$

Tabelle 4.3: p-te Kreiskörper, Relativklassenzahlen und größter Primteiler

4.4.2 Hinreichend große Primfaktoren in der Klassenzahl

In diesem Abschnitt zeigen wir, dass die Klassenzahl der in Abschnitt 4.3 für die NF-Kryptographie vorgeschlagenen Klassengruppen und Zahlkörper hinreichend große Primfaktoren enthält. Dieses ist von entscheidender Bedeutung, um den Erfolg eines Pohlig-Hellman-Angriffes auf die NF-Kryptoverfahren zu verhindern (siehe Abschnitte 4.2, 4.1.2).

Am Anfang dieses Kapitels haben wir erklärt, wann wir eine ganze Zahl B-glatt nennen für eine positive Zahl B. Bislang gab es weder theoretische noch praktische Untersuchungen über die Glattheitswahrscheinlichkeit von Klassenzahlen algebraischer Zahlkörper vom Grade größer als 2. Hamdy schätzt in [Ham02, Kap.5] und [HM00] die Wahrscheinlichkeit ab, dass die Klassenzahl eines imaginär-quadratischen Zahlkörpers B-glatt ist. Hamdys Analyse basiert allerdings auf einigen wenig plausiblen Annahmen. Die so durch Hamdy erhaltene Abschätzung für die Glattheitswahrscheinlichkeit von Klassenzahlen stimmt nicht sonderlich gut mit experimentellen Resultaten überein (siehe [Ham02, S.63]).

In einem ersten Unterabschnitt zeigen wir unter Verwendung der Cohen-Martinet-Heuristik [CM87, CM90], dass Klassenzahlen mit ähnlicher Wahrscheinlichkeit B-glatt sind wie Zufallszahlen (aus der Größenordnung der Klassenzahlen). Für die Wahrscheinlichkeit, dass eine Zufallszahl B-glatt ist, kennt man eine gute Approximation. Dies führt zu einer Abschätzung für die Glattheitswahrscheinlichkeit von Klassenzahlen. Diese Wahrscheinlichkeit scheint gemäß unserer Analyse hinreichend klein zu sein.

Unsere theoretischen Argumente basieren auf der Gültigkeit einiger unbewiesener[3] Annahmen. Daher untersuchen wir die Glattheitswahrscheinlichkeit von Klassenzahlen in einem zweiten Unterabschnitt auch experimentell. Wir werden sehen, dass die praktischen Resultate hervorragend mit den zuvor getroffenen Abschätzungen übereinstimmen.

4.4.2.1 Theoretischer Nachweis

Sei $S \in \mathbb{Z}_{>0}$ beliebig. Wir betrachten die Menge aller kubischen Stender-Zahlkörper $K = \mathbb{Q}(\sqrt[3]{D^3 \pm 1})$, $D \in \mathbb{Z}_{>16}$ mit Diskriminante Δ, für die $|\Delta| \leqslant S$ gilt. Aus [Coh95, Theorem 6.4.13] folgt: Diese Menge ist endlich. Aus dieser Menge wählen wir einen Zahlkörper K zufällig aus.

[3] obgleich plausibler

In diesem Unterabschnitt werden wir folgendes zeigen: Die Wahrscheinlichkeit, bei in diesem Sinne zufälliger Wahl eines kubischen Stender-Körpers K eine sehr glatte Klassenzahl (d.h. eine Klassenzahl mit ausschließlich kleinen Primteilern) zu erhalten, ist sehr gering. Unsere Analyse basiert auf geeigneten Annahmen, die wir weiter unten schildern. Die wichtigste Annahme ist, dass die von Cohen und Martinet in [CM90] über algebraische Zahlkörper getroffenen Hypothesen für kubische Stender-Körper korrekt sind. Dann nämlich scheint die Cohen-Martinet-Heuristik [CM87] für komplex-kubische Zahlkörper und damit für die kubischen Stender-Körper zu gelten. Für Primzahlen p verwenden wir folgende Bezeichnungen:

$(p)_\alpha$	$= \prod_{1 \leqslant k \leqslant \alpha} \left(1 - p^{-k}\right)$, p prim
$(p)_\infty$	$= \prod_{k \geqslant 1} \left(1 - p^{-k}\right)$, p prim
H_K	der *Nicht-3-Anteil von* $Cl(K)$: Untergruppe von $Cl(K)$, die alle Elemente mit Ordnungen z enthält, wobei $z \in \mathbb{Z}$ beliebig mit $3 \nmid z$.

Kubische Stender-Körper sind komplex-kubische Zahlkörper (d.h. es gibt $r_1 = 1$ reelle Einbettung, $2r_2 = 2$ komplexe Einbettungen in den Körper der komplexen Zahlen). Wir nehmen an, dass die Heuristik von Cohen und Martinet [CM87] korrekt ist. Diese besagt in unserem Kontext insbesondere folgendes für beliebige ganze Zahlen m mit $3 \nmid m$:

Annahme 4.4.1 (Cohen-Martinet)

$$\Pr\left[\ m \mid |H_K|\ \right] = \prod_{p^\alpha \| m} \left(\left(1 - \frac{(p)_\infty}{(p)_1}\right) \cdot \sum_{0 \leqslant \beta < \alpha} \left((p^{2\beta} \cdot (p)_\beta)^{-1} \right) \right) \tag{4.11}$$

Zudem nehmen wir an: Für zwei verschiedene Primzahlen $p_1, p_2 \neq 3$ ist die Wahrscheinlichkeit, dass p_1 die Klassenzahl teilt, unabhängig von der Wahrscheinlichkeit, dass p_2 die Klassenzahl teilt. Formal treffen wir folgende Annahme:

Annahme 4.4.2
Sei $S \in \mathbb{Z}_{>0}$ beliebig. Wir betrachten die Menge aller kubischen Zahlkörper $K = \mathbb{Q}(\sqrt[3]{D^3 \pm 1})$ mit Diskriminante Δ, für die $|\Delta| \leqslant S$ gilt. Aus dieser Menge wählen wir einen Zahlkörper K zufällig aus. Für $i \in \{1, \ldots, k\}, k \in \mathbb{Z}_{>0}$ seien $p_i \neq 3$ beliebige paarweise verschiedene Primzahlen. Dann gilt:

$$\lim_{S \to \infty} \Pr\left[\prod_{1 \leqslant i \leqslant k} p_i \mid h(K) \right] = \lim_{S \to \infty} \prod_{1 \leqslant i \leqslant k} \Pr\left[\ p_i \mid h(K)\ \right] . \tag{4.12}$$

Unser zentrales Resultat in diesem Unterabschnitt lautet:

Satz 4.4.3
Sei $S \in \mathbb{Z}_{>0}$ eine Schranke. Sei $\rho(u) \overset{\mathbb{R}_{>0}}{=} \frac{1}{u} \int_{u-1}^{u} \rho(t)\,dt$.[4] Für einen zufällig gewählten kubischen Stender-Körper K mit Diskriminante Δ, $|\Delta| \leqslant S$ und die Glattheitsschranke $B = S^{\frac{0.45}{u}}$ mit

[4] Dickmannsche ρ-Funktion. Es gilt $\rho(u) = u^{-u+o(1)}$.

$1 \leqslant u \leqslant exp((\ln B)^{\frac{3}{5}-\epsilon})$, $\epsilon > 0$, *gilt*

$$\begin{aligned} \rho(u)\left(1+O_\epsilon\left(\frac{\ln(u+1)}{\ln B}\right)\right) &< \Pr\left[\ h(K)\ \textit{ist } B\textit{-glatt}\ \right] \\ &\leqslant \left(u+\frac{e^\gamma u^2}{0.45}\cdot\frac{1}{\ln S}\right)\cdot\rho(u)\left(1+O_\epsilon\left(\frac{\ln(u+1)}{\ln B}\right)\right)\ . \end{aligned} \tag{4.13}$$

Zum Beweis des Satzes gehen wir folgendermaßen vor:

1. Wir wenden die Cohen-Martinet-Heuristik [CM87] auf kubische Stender-Körper an. In [CM87] finden sich nur Aussagen (in wenig konstruktiver Form) über Teilbarkeitseigenschaften der Ordnung der Untergruppe H_K von $Cl(K)$, während wir an Teilbarkeitseigenschaften von $h(K) = |Cl(K)|$ (in konstruktiver Form) interessiert sind. Wir weisen daher zunächst nach, dass für $h(K) = |Cl(K)|$ abgesehen vom Teiler 3 exakt die gleichen Teilbarkeitseigenschaften gelten wie für $|H_K|$.

2. Dann setzen wir die Wahrscheinlichkeit, dass die Klassenzahl $h(K)$ B-glatt ist, in Bezug zur Wahrscheinlichkeit, dass eine Zufallszahl x aus der Größenordnung von $h(K)$ B-glatt ist.

3. Anschließend zitieren wir die aus der Literatur bekannte Wahrscheinlichkeit, dass Zufallszahlen B-glatt sind.

4. Die Kombination von 2. und 3. liefert uns die Wahrscheinlichkeit (in geschlossener Form), dass die Klassenzahl B-glatt ist, d.h. nur Primteiler unterhalb einer Schranke B besitzt.

Aus Annahme 4.4.1 folgt für Primzahlen $p \neq 3$:

$$\Pr\left[\ p \mid |H_K|\ \right] = 1 - \frac{(p)_\infty}{(p)_1} \tag{4.14}$$

Hier werden Aussagen über Teilbarkeitseigenschaften der Ordnung der Untergruppe H_K von $Cl(K)$ gemacht, während wir an Teilbarkeitseigenschaften von $h(K) = |Cl(K)|$ interessiert sind. Es ist $H_K = Cl(K)$, falls $3 \nmid h(K) = |Cl(K)|$; ansonsten ist $H_K \neq Cl(K)$. Wir zeigen:

Lemma 4.4.4
Eine ganze Zahl m, die kein Vielfaches von 3 ist, teilt die Klassenzahl $h(K)$ genau dann, wenn m die Ordnung $|H_K|$ teilt. Die Gleichungen (4.11) und (4.14) gelten demnach unter der Annahme der Gültigkeit der Cohen-Martinet-Heuristik für komplex-kubische Zahlkörper entsprechend für die Klassenzahlen $h(K)$.

Beweis: Sei S_3 die Menge aller Elemente von $Cl(K)$, deren Ordnung eine Potenz von 3 ist (einschließlich $3^0 = 1$). S_3 selbst ist dann eine Untergruppe von $Cl(K)$ mit Ordnung 3^r, $r \in \mathbb{Z}_{\geqslant 0}$. ($S_3$ heißt *3-Sylow-Gruppe* oder *3-Anteil* von $Cl(K)$.) Als endliche Abelsche Gruppe lässt sich $Cl(K)$ schreiben als direktes inneres Produkt $Cl(K) = S_3 H_K$, wobei H_K genau die Elemente mit zu 3 teilerfremder Ordnung enthält.[5] Wegen $|S_3| = 3^r$ ist

$$h(K) = |Cl(K)| = 3^r \cdot |H_K|\ .$$

[5] Es gilt $3 \nmid |H_K|$.

Hieraus folgt unmittelbar die Behauptung. □

Wir werden später die Wahrscheinlichkeit aus (4.14) nach oben abschätzen müssen. Wir beweisen folgendes

Lemma 4.4.5
Für Primzahlen p gilt:

$$1 - \frac{(p)_\infty}{(p)_1} < 1 - (p)_\infty < \frac{1}{p} \cdot \left(1 + \frac{1}{p}\right)$$
$$\text{und} \qquad (p)_\infty > 1 - \frac{1}{p} \cdot \left(1 + \frac{1}{p}\right)$$

Beweis: Die spezielle unendliche Eulersche Produktreihe $\prod_{k=1}^{\infty}\left(1-x^k\right)$ lässt sich als Potenzreihe darstellen und konvergiert für alle reellen Zahlen x mit $|x| < 1$ ([Sch91, S. 386]):

$$\prod_{k=1}^{\infty}\left(1-x^k\right) = 1 + \sum_{n=1}^{\infty}(-1)^n \cdot \left[x^{\frac{n}{2}\cdot(3n+1)} + x^{\frac{n}{2}\cdot(3n-1)}\right] \quad . \tag{4.15}$$

Für Primzahlen p erhalten wir mit $0 < x := \frac{1}{p} < 1$:

$$\begin{aligned}
1 - \frac{(p)_\infty}{1-\frac{1}{p}} &< 1 - (p)_\infty \\
&= \sum_{n=1}^{\infty}(-1)^{n+1} \cdot \left[\left(\frac{1}{p}\right)^{\frac{n}{2}\cdot(3n+1)} + \left(\frac{1}{p}\right)^{\frac{n}{2}\cdot(3n-1)}\right] \\
&= \underbrace{\left(\frac{1}{p^2}+\frac{1}{p}\right)}_{n=1} \underbrace{- \left(\frac{1}{p^7}+\frac{1}{p^5}\right)}_{n=2} \underbrace{+ \left(\frac{1}{p^{15}}+\frac{1}{p^{12}}\right)}_{n=3} \underbrace{- \left(\frac{1}{p^{26}}+\frac{1}{p^{22}}\right)}_{n=4} \underbrace{+ \left(\frac{1}{p^{40}}+\frac{1}{p^{35}}\right)}_{n=5} - + \dots \\
&= \left(\frac{1}{p^2}+\frac{1}{p}\right) - \underbrace{\left[\underbrace{\left(\frac{1}{p^7}+\frac{1}{p^5}\right) - \left(\frac{1}{p^{15}}+\frac{1}{p^{12}}\right)}_{>0} + \underbrace{\left(\frac{1}{p^{26}}+\frac{1}{p^{22}}\right) - \left(\frac{1}{p^{40}}+\frac{1}{p^{35}}\right)}_{>0} + - \dots\right]}_{>0} \\
&< \frac{1}{p^2}+\frac{1}{p} = \frac{1}{p} \cdot \left(1 + \frac{1}{p}\right)
\end{aligned}$$

Die zweite Behauptung ergibt sich durch einfache Umformung der ersten Behauptung. □

Vergleich der Teilbarkeitseigenschaften von Klassenzahlen und Zufallszahlen
Wir fixieren eine beliebige Schranke $S \in \mathbb{Z}_{>0}$. Wir betrachten die Menge aller kubischen Zahlkörper $K = \mathbb{Q}(\sqrt[3]{D^3 \pm 1})$ mit Diskriminante Δ, für die $|\Delta| \leqslant S$ gilt. Wir wählen aus dieser Menge zufällig einen Zahlkörper K aus. Dann fixieren wir eine zufällige Glattheitsschranke $B \in \mathbb{Z}_{>0}$ mit $B < |\Delta|^{0.45}$. Nun vergleichen wir die Wahrscheinlichkeit, dass die Klassenzahl $h(K)$ bei in diesem Sinne zufälliger Wahl eines kubischen Zahlkörpers $K = \mathbb{Q}(\sqrt[3]{D^3 \pm 1})$ B-glatt ist, mit der Wahrscheinlichkeit, dass eine aus der Größenordnung von $h(K)$ gewählte Zufallszahl $x \in \mathbb{Z}_{>0}$

B-glatt ist. Die Schranke S lassen wir gegen unendlich gehen. Unsere Experimente zeigen (siehe Abschnitt 4.4.2.2), dass wir die Klassenzahl in der Größenordnung $|\Delta|^{0.45} \leqslant S^{0.45}$ erwarten dürfen. Daher wählen wir die Zufallszahl x zufällig aus $\{1, \ldots, \lfloor S^{0.45} \rfloor\}$.

Für von S abhängige positive reelle Zahlen $c_1(S), c_2(S)$ sei

$$c_1(S) < \frac{\Pr[\ h(K) \text{ ist } B\text{-glatt }]}{\Pr[\ x \text{ ist } B\text{-glatt }]} \leqslant c_2(S)\ . \tag{4.16}$$

Mit der Kenntnis geeigneter $c_1(S), c_2(S)$ können wir die Wahrscheinlichkeit abschätzen, dass die Klassenzahl $h(K)$ B-glatt ist; denn wir können die Glattheitswahrscheinlichkeit für zufällig gewählte Zahlen x gut approximieren, wie wir später sehen werden.

Lemma 4.4.6
In obigem Kontext gilt:

$$c_1(S) = 1,\ c_2(S) = u,\ \textit{d.h. } 1 < \frac{\Pr[\ h(K) \textit{ ist } B\textit{-glatt }]}{\Pr[\ x \textit{ ist } B\textit{-glatt }]} \leqslant u\ . \tag{4.17}$$

Beweis: Wir wählen eine Glattheitsschranke $B := S^{\frac{0.45}{u}}$, deren Größe wir über eine reelle Zahl $u \geqslant 1$ steuern. Dann gilt:

$$\begin{aligned}
\lim_{S\to\infty} \frac{\Pr[\ h(K) \text{ ist } B\text{-glatt }]}{\Pr[\ x \text{ ist } B\text{-glatt }]} &= \lim_{S\to\infty} \frac{\Pr[\ p \nmid h(K) \text{ für alle primen } p > B\ ]}{\Pr[\ p \nmid x \text{ für alle primen } p > B\ ]} \\
&= \lim_{S\to\infty} \frac{\prod_{B<p\leqslant S^{0.45}} \Pr[\ p \nmid h(K)\ ]}{\prod_{B<p\leqslant S^{0.45}} \Pr[\ p \nmid x\ ]} \\
&= \lim_{S\to\infty} \frac{\prod_{B<p\leqslant S^{0.45}} (1 - \Pr[\ p \mid h(K)\ ])}{\prod_{B<p\leqslant S^{0.45}} (1 - \Pr[\ p \mid x\ ])} \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{1 - \Pr[\ p \mid h(K)\ ]}{1 - \Pr[\ p \mid x\ ]} \\
&\overset{(4.14)}{=} \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{\frac{(p)_\infty}{1-\frac{1}{p}}}{1 - \frac{1}{p}} \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{(p)_\infty}{\left(1 - \frac{1}{p}\right)^2}
\end{aligned} \tag{4.18}$$

Mit Hilfe von (4.18) und Lemma 4.4.5 bestimmen wir eine geeignete Zahl $c_1(S)$:

$$\begin{aligned}
\lim_{S\to\infty} \frac{\Pr[\ h(K) \text{ ist } B\text{-glatt }]}{\Pr[\ x \text{ ist } B\text{-glatt }]} &= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{(p)_\infty}{\left(1-\frac{1}{p}\right)^2} \\
&\overset{Le.4.4.5}{>} \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{1-\frac{1}{p}\cdot\left(1+\frac{1}{p}\right)}{\left(1-\frac{1}{p}\right)^2} \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{1-\frac{1}{p}-\frac{1}{p^2}}{\left(1-\frac{1}{p}\right)^2} \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \left[\frac{1}{1-\frac{1}{p}} - \frac{1}{p^2\cdot\frac{(p-1)^2}{p^2}}\right] \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{p(p-1)-1}{(p-1)^2} \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{p^2-p-1}{p^2-2p+1} \\
&= \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \underbrace{\left(1+\frac{p-2}{(p-1)^2}\right)}_{>1} \\
&> 1 =: c_1(S) \qquad (4.19)
\end{aligned}$$

Wegen $\frac{(p)_\infty}{(p)_1} = \prod_{k\geqslant 2}\underbrace{\left(1-\frac{1}{p^k}\right)}_{<1,\ >0} < 1$ ist $\frac{(p)_\infty}{((p)_1)^2} < \frac{1}{(p)_1} = \frac{1}{1-\frac{1}{p}}$.

Durch Einsetzen in (4.18) können wir eine geeignete Zahl $c_2(S)$ bestimmen:

$$\begin{aligned}
\lim_{S\to\infty} \frac{\Pr[\ h(K) \text{ ist } B\text{-glatt }]}{\Pr[\ x \text{ ist } B\text{-glatt }]} &< \lim_{S\to\infty} \prod_{B<p\leqslant S^{0.45}} \frac{1}{1-\frac{1}{p}} \\
&\overset{(B=S^{\frac{0.45}{u}})}{=} \lim_{S\to\infty} \frac{1}{\frac{\prod_{p\leqslant S^{0.45}}\left(1-\frac{1}{p}\right)}{\prod_{p\leqslant S^{\frac{0.45}{u}}}\left(1-\frac{1}{p}\right)}} \\
&= \lim_{S\to\infty} \frac{\prod_{p\leqslant S^{\frac{0.45}{u}}}\left(1-\frac{1}{p}\right)}{\prod_{p\leqslant S^{0.45}}\left(1-\frac{1}{p}\right)} \\
&\overset{(4.21)}{=} \lim_{S\to\infty} \frac{\frac{1}{e^\gamma \ln S^{\frac{0.45}{u}}}}{\frac{1}{e^\gamma \ln S^{0.45}}} \\
&= \lim_{S\to\infty} \frac{e^\gamma\cdot 0.45\cdot\ln S}{e^\gamma\cdot\frac{0.45}{u}\cdot\ln S} \\
&= u =: c_2(S) \qquad (4.20)
\end{aligned}$$

Hierbei besagt das Theorem von Mertens für eine positive Zahl t (siehe [Ros94, Kap.12]):

$$\prod_{p \text{ prim},\, p \leqslant t} \left(1 - \frac{1}{p}\right) = \frac{1}{e^{\gamma} \ln t} + O\left(\frac{1}{\ln^2 t}\right) \ . \tag{4.21}$$

γ bezeichnet die Eulersche Konstante: $\gamma = \lim_{N\to\infty} \left[\left(\sum_{n=1}^{N} \frac{1}{n}\right) - \ln N\right] = 0.5772156649\ldots$ ([HW79]). Es ist $e^{\gamma} \approx 1.781$. Bei der obigen Anwendung des Mertens-Theorems vernachlässigen wir den zweiten Summanden, weil dieser asymptotisch keine Rolle spielt. □

Illustration des Zusammenhangs zwischen Teilbarkeitseigenschaften von Klassenzahlen und Zufallszahlen:
Für $|\Delta| \leqslant S = 2^{404}$, d.h. die Schlüssellänge eines NF-Kryptoverfahrens wäre 404 Bits[6], führt Lemma 4.4.6 für die Glattheitsschranke $B = S^{\frac{0.45}{2}} \approx 2^{91}$ beispielsweise zu

$$\Pr\left[\ x \text{ ist } B\text{-glatt } \right] < \Pr\left[\ h(K) \text{ ist } B\text{-glatt } \right] \leqslant 2 \cdot \Pr\left[\ x \text{ ist } B\text{-glatt } \right] \ .$$

Wählen wir die Schranke S mindestens in der Größenordnung 2^{404}, so ist für die Glattheitsschranke $B = S^{\frac{0.45}{u}}$ die Wahrscheinlichkeit dafür, dass $h(K)$ B-glatt ist, höchstens u mal so groß wie die Wahrscheinlichkeit, dass x B-glatt ist.

Wir beweisen nun Satz 4.4.3.

Beweis: Die Wahrscheinlichkeit, dass eine Zufallszahl $x \leqslant S^{0.45}$ B-glatt ist, können wir gemäß [HS97] durch die Dickmannsche ρ-Funktion[7] approximieren. Es ist

$$\rho(u) = \frac{1}{u} \int_{u-1}^{u} \rho(t)\, dt$$

Mit $x \leqslant S^{0.45}$, und der Glattheitsschranke $B = S^{\frac{0.45}{u}}$ (d.h. $u = \frac{\ln S^{0.45}}{\ln B}$) für ein reelles $u > 0$ gilt ([HS97, Theorem 1])

$$\Pr\left[\ x \text{ ist } B\text{-glatt } \right] = \rho(u) \left(1 + O_{\epsilon}\left(\frac{\ln(u+1)}{\ln B}\right)\right) \ . \tag{4.22}$$

Hierbei ist $B \geqslant 2$, $\epsilon > 0$ und $1 \leqslant u \leqslant \exp((\ln B)^{3/5-\epsilon})$; die O-Konstante ist nur von ϵ abhängig. Der Audruck $\frac{\ln(u+1)}{\ln B}$ ist positiv und für große Glattheitsschranken B fast 0. Wir vernachlässigen diesen Ausdruck daher. Durch Einsetzen von (4.22) in Lemma 4.4.6 erhalten wir die Behauptung. □

Für betragsmäßig sehr große Diskriminanten $|\Delta| \leqslant S$ und die Glattheitsschranke $B = \sqrt[u]{h(K)}$ ergibt sich mit der Annahme, dass $\rho(u) \approx u^{-u}$, etwa folgende Abschätzung:

$$\frac{1}{u^u} < \Pr\left[\ h(K) \text{ ist } B\text{-glatt } \right] \leqslant \frac{1}{u^{u-1}} \ . \tag{4.23}$$

Analog kann man vorgehen, um die Glattheitswahrscheinlichkeit von Klassenzahlen der anderen in Abschnitt 4.3 für die NF-Kryptographie vorgeschlagenen algebraischen Zahlkörper abzuschätzen. Man hat hier jeweils mit der (4.11) entsprechenden Formel für den betreffenden Zahlkörper aus der Cohen-Martinet-Heuristik [CM87] zu starten.

[6] entspricht der Sicherheit von RSA-1024 Bit

[7] Es gilt $\rho(u) = u^{-u+o(1)}$

Tabelle 4.4: Diskriminanten kubischer Stender-Körper mit $|\Delta|^{\frac{0.45}{u}}$-glatter Klassenzahl, wobei $2^{124} < |\Delta| < 2^{169}$

u	Anteil	$\rho(u)$	$\left[\rho(u),\ \left(u + \frac{e^\gamma u^2}{0.45} \cdot \frac{1}{\ln S}\right)\rho(u)\right]$
1.5	0.70634	0.59453	[0.59453, 0.91070]
2.0	0.46825	0.30685	[0.30685, 0.63105]
2.5	0.25397	0.13033	[0.13033, 0.33734]
3.0	0.07937	0.04861	[0.04861, 0.15201]
3.5	0.02381	0.01623	[0.01623, 0.05962]
4.0	0	0.00491	[0.00491, 0.02075]
4.5	0	0.00137	[0.00137, 0.00656]
5.0	0	0.00035	[0.00035, 0.00187]
5.5	0	0.00009	[0.00009, 0.00053]
6.0	0	0.00002	[0.00002, 0.00013]
6.5	0	0.00000	[0.00000, 0.00000]

4.4.2.2 Experimenteller Nachweis

Wir haben die Klassenzahlen und deren Primfaktorzerlegung von kubischen Stender-Körpern mit 1479 zufällig gewählten Diskriminanten aus dem Bereich $2^{65} \leqslant |\Delta| \leqslant 2^{169}$ berechnet (siehe Anhang). Hiervon waren 203 Diskriminanten betragsmäßig größer als 2^{124}. Die Tabellen 4.4 und 4.5 zeigen für verschiedene Werte von u (erste Spalte) den Anteil der Diskriminanten mit $B = |\Delta|^{\frac{0.45}{u}}$-glatter Klassenzahl (zweite Spalte). Die dritte und vierte Spalte dienen zum Vergleich unserer experimentellen Resultate mit den im vorangegangenen Unterabschnitt hergeleiteten Schranken für die Glattheitswahrscheinlichkeit von Klassenzahlen: Spalte 3 zeigt den Wert der Dickmannschen ρ-Funktion als Approximation für die Wahrscheinlichkeit, dass eine zufällig in der gleichen Größenordnung gewählte ganze Zahl x B-glatt ist; Spalte 4 gibt das theoretisch vorausgesagte Intervall für die Glattheitswahrscheinlichkeit von Klassenzahlen an. Tabelle 4.4 beschränkt sich auf Stender-Körper mit Diskriminanten Δ mit $2^{124} < |\Delta| < 2^{169}$. Tabelle 4.5 berücksichtigt alle untersuchten Stender-Körper mit Diskriminanten Δ mit $2^{65} < |\Delta| < 2^{169}$, wobei hier der Großteil der Diskriminanten (86 %) betragsmäßig kleiner als 2^{124} gewählt wurde. In beiden Tabellen verwenden wir in der vierten Spalte die Schranke $S = 2^{404}$, so dass diese Spalte jeweils auch die Glattheitswahrscheinlichkeit von Klassenzahlen von Stender-Körpern mit Diskriminanten der Größenordnung -2^{404} voraussagt, welche real in der Kryptographie zum Einsatz kommen könnten. Das Verwenden einer exakteren Schranke S für die tatsächlich in den Experimenten verwendeten Diskriminanten würde nur zu marginalen Vergrößerungen der oberen Intervallgrenzen in der vierten Spalte führen; die obere Intervallgrenze hat etwa den Wert $u \cdot \rho(u)$.

Die Tabellen zeigen:

- Die experimentell ermittelten Glattheitswahrscheinlichkeiten von Klassenzahlen kubischer Stender-Körper stimmen in Tabelle 4.5 bereits recht gut mit den theoretisch vorausgesagten Intervallen überein; denn der Anteil B-glatter Klassenzahlen liegt für $u < 3.5$ innerhalb des vorausgesagten Intervalls und für $3.5 \leqslant u$ unmittelbar oberhalb der rechten Intervallgrenze. (Der Anteil 0 für die B-glatten Klassenzahlen mit $u = 6.0$ stimmt ebenfalls mit der

Tabelle 4.5: Diskriminanten kubischer Stender-Körper mit $|\Delta|^{\frac{0.45}{u}}$-glatter Klassenzahl, wobei $2^{65} < |\Delta| < 2^{169}$

u	Anteil	$\rho(u)$	$\left[\rho(u),\ \left(u + \frac{e^\gamma u^2}{0.45} \cdot \frac{1}{\ln S}\right)\rho(u)\right]$
1.5	0.79716	0.59453	[0.59453, 0.91070]
2.0	0.50372	0.30685	[0.30685, 0.63105]
2.5	0.28330	0.13033	[0.13033, 0.33734]
3.0	0.14266	0.04861	[0.04861, 0.15201]
3.5	0.06085	0.01623	[0.01623, 0.05962]
4.0	0.02231	0.00491	[0.00491, 0.02075]
4.5	0.01014	0.00137	[0.00137, 0.00656]
5.0	0.00406	0.00035	[0.00035, 0.00187]
5.5	0.00068	0.00009	[0.00009, 0.00053]
6.0	0	0.00002	[0.00002, 0.00013]
6.5	0	0.00000	[0.00000, 0.00000]

Voraussage überein, da wir 1479 Datenpunkte betrachten und $0.00013 \cdot 1479 < 0.5$.)

- Die für betragsmäßig größere Diskriminanten ($2^{124} < |\Delta| < 2^{169}$) experimentell bestimmten Glattheitswahrscheinlichkeiten liegen für alle Glattheitsschranken B (d.h. für alle Werte u) innerhalb der theoretisch vorausgesagten Intervalle. (Der Anteil 0 für die B-glatten Klassenzahlen mit $u \geqslant 4.0$ stimmt ebenfalls mit der Voraussage überein, da wir 203 Datenpunkte betrachten und $0.00491 \cdot 203 < 0.5$.)

- Der Vergleich der zweiten Spalten der Tabellen 4.4,4.5 zeigt, dass die (relative) Glattheitswahrscheinlichkeit von Klassenzahlen mit zunehmender Größe des Absolutbetrages der Diskriminante abnimmt. Dies lässt darauf schließen, dass sich Klassenzahlen von für die Kryptographie relevanten Diskriminanten ($\approx -2^{404}$) bzgl. ihrer Glattheitswahrscheinlichkeit tatsächlich noch viel günstiger verhalten als in unseren Experimenten. In der Tat scheinen Klassenzahlen von Stender-Körpern mit betragsmäßig großer Diskriminante mit ähnlicher Wahrscheinlichkeit B-glatt zu sein wie Zufallszahlen der gleichen Größenordnung.

Bemerkung zu p-ten Kreiskörpern p-te Kreiskörper bilden eine Ausnahme bei unseren Betrachtungen, da hier jeweils explizit die Existenz eines hinreichend großen Primteilers der Klassenzahl nachgewiesen werden kann. Für Details verweisen wir auf Abschnitt 4.3.

4.5 Auswahl kryptographisch relevanter Schlüssellängen (Wahl der Diskriminante)

In Abschnitt 4.1 haben wir Buchmanns Klassengruppenalgorithmus [Buc89] als den effizientesten allgemeinen Angriff auf die NF-Kryptosysteme identifiziert. Zudem haben wir für kleine Diskriminantenlängen die tatsächliche Laufzeit dieses Algorithmus gemessen. In diesem Abschnitt extrapolieren wir zunächst die gemessenen Laufzeiten, um die Schlüssellängen zu bestimmen, die

zum Erreichen des gewünschten Sicherheitslevels verwendet werden müssen. Anschließend werden wir unsere Schlüssellängen für NF-Kryptosysteme mit Schlüssellängen traditioneller Public-Key-Kryptosysteme vergleichen. Hierzu wenden wir das weithin anerkannte Modell [LV01] von A. Lenstra und Verheul auf die NF-Kryptosysteme an.

Extrapolation

Für gegebene Diskriminante Δ müssen $L(|\Delta|)[\frac{1}{2}, \sqrt{2}+o(1)]$ viele Bitoperationen ausgeführt werden, um das Berechnungsproblem zu lösen (z.B. Berechnung eines Vielfachen der Klassenzahl $h(\Delta)$), um also die NF-Kryptosysteme aus Kapitel 3 zu brechen.

Die Laufzeiten des schnellsten Algorithmus für das Berechnungsproblem können bei Verwendung von Diskriminanten in kryptographisch relevanter Größenordnung nicht praktisch gemessen werden. Um die Laufzeiten vorherzusagen und mit Laufzeiten zum Brechen traditioneller Kryptoverfahren vergleichbar zu machen, gehen wir vor, wie in vergleichbaren Situationen üblich (siehe z.B. [LV01], [Ham02]):

- Wir stellen fest, dass für variierende Eingabegrößen (Diskriminanten variieren, Zahlkörpergrad fix) der Quotient aus Anzahl durchzuführender Bitoperationen und Laufzeit auf einem Computer konstant ist:

$$\frac{L(|\Delta_1|)[\frac{1}{2}, \sqrt{2}]}{t_{\Delta_1}} = \frac{L(|\Delta_2|)[\frac{1}{2}, \sqrt{2}]}{t_{\Delta_2}} =: c \tag{4.24}$$

- Wir ignorieren in der subexponentiellen Laufzeitfunktion aus (4.24) den o(1)-Term.

- Wir berechnen die Konstante c bei fixem Zahlkörpergrad als arithmetisches Mittel der Quotienten aus (4.24), wobei wir die Diskriminantenlängen so klein wählen, dass wir die tatsächlichen Laufzeiten auf einem Computer messen können (siehe Anhang B). Mit Hilfe der PARI/GP-Implementierung [BBB+02] des Buchmann-Algorithmus [Buc89] haben wir die Klassenzahlen von Stender-Körpern $K = \mathbb{Q}(\omega)$ mit $\omega = \sqrt[n]{D^n - 1}$ der Grade $n \in \{3, 4\}$ bestimmt $(D \in \mathbb{Z}_{>3})$. Die Laufzeit des Algorithmus wurde dabei jeweils aus Sun-Solaris-Workstations mit UltraSparc II-Prozessoren und 450 MHz Taktfrequenz gemessen. Wir haben bei unseren Experimenten als Stackgröße 900 MB realen physischen Speicher verwendet. Bei Laufzeitmessungen in 2500 Stender-Körpern vom Grad 3 ergibt sich somit:

$$c = 1.5 \cdot 10^9$$

Bei Laufzeitmessungen in 110 Stender-Körpern vom Grad 4 erhalten wir:

$$c = 2.4 \cdot 10^8$$

Die aus (4.24) abgeleitete Formel

$$t_{\Delta_2} = \frac{L(|\Delta_2|)[\frac{1}{2}, \sqrt{2}]}{c} \tag{4.25}$$

liefert uns die in Tabelle 4.6 angegebenen extrapolierten Laufzeiten: Für die Diskriminantenlänge aus Spalte 1 enthält Spalte 2 (bzw. Spalte 4) die auf einer 450 MHz-UltraSparc II-Workstation erwartete Laufzeit in Jahren bei Verwendung eines Stender-Körpers vom Grad 3 (bzw. vom Grad 4).

- Wir wollen die prognostizierten Laufzeiten rechnerunabhängig ausdrücken, um den Aufwand zum Brechen von NF-Kryptoverfahren vergleichbar mit dem Aufwand zum Brechen traditioneller Public-Key-Kryptoverfahren zu machen. Hierzu verwenden wir wie in [LV01] die Maßeinheit 1 MIPS für die Rechengeschwindigkeit eines Computers oder einer CPU. Eine 1 MIPS-CPU führt eine Million Operationen pro Sekunde aus. In der Arbeit [HM00] wurde per Benchmark ermittelt, dass eine Sun-Solaris-Workstation mit UltraSparc I-Prozessor mit 170MHz Taktfrequenz 100 MIPS schnell ist (d.h. 100 Millionen Operationen pro Sekunde ausführt). Daher erscheint es vernünftig – pessimistisch aus der Sicht der Benutzer der NF-Kryptosysteme – anzunehmen, dass unsere UltraSparc II-Prozessoren 400 MIPS schnell sind, d.h. sie führen 400 Millionen Operationen pro Sekunde aus. Die Laufzeit eines Algorithmus drücken wir rechnerunabhängig in MIPS-Jahren aus. Ein MIPS-Jahr ist die Anzahl der Operationen, die eine 1 MIPS-CPU in einem Jahr ausführen kann, also $3.1535 \cdot 10^{13}$ Operationen. Der Einfachheit halber nehmen wir an, eine Operation entspreche einer Bitoperation.

  Die Anzahl der MIPS-Jahre ($\#\mathrm{MY}(|\Delta|)$), die zum Brechen eines NF-Kryptosystems durch Lösen des zugrunde liegenden Berechnungsproblems bei gegebener Diskriminante Δ investiert werden muss, berechnen wir wie folgt (in [Ham02] wird implizit die gleiche Formel verwendet):

$$\#\mathrm{MY}(|\Delta|) = t_\Delta \cdot \frac{< \mathrm{MIPS - Rate\ PC} >}{1\mathrm{MY}} = t_\Delta \cdot \frac{400 \cdot 10^6 s^{-1}}{3.1535 \cdot 10^{13}} \qquad (4.26)$$

  Hierbei ist t_Δ die auf einem Computersystem erwartete Laufzeit des Angriffes gemäß Gleichung (4.25). Als MIPS-Rate desselben Computersystems haben wir die unserem Computer entsprechenden 400 MIPS, d.h. $400 \cdot 10^6$ Operationen pro Sekunde, eingesetzt. In Tabelle 4.6 enthält Spalte 3 (bzw. Spalte 5) den aus dieser Formel resultierenden Berechnungsaufwand bei Verwendung von Stender-Körpern vom Grad 3 (bzw. vom Grad 4) mit entsprechender Diskriminantenlänge.

Anwendung des Modells von Lenstra und Verheul

Wir vergleichen nun die zum Brechen der NF-Kryptosysteme notwendige Zeit mit der Zeit, die zum Brechen heute in der Praxis verwendeter Public-Key-Kryptosysteme notwendig ist.

A. Lenstra und Verheul entwickelten in ihrer Arbeit [LV01] ein weit akzeptiertes Modell, um Mindestschlüssellängen von Kryptosystemen anzugeben, die in der Zukunft ein gewünschtes Sicherheitsniveau gewährleisten. Die Arbeit setzt Schlüssellängen von symmetrischen Kryptoverfahren mit denen von Public-Key-Verfahren wie RSA oder Elliptischen Kurven-Verfahren sicherheitstechnisch in Bezug.

Wir wenden das Modell auf NF-Kryptosysteme an und machen gemäß [LV01] folgende Annahmen:

1. Das symmetrische Kryptoverfahren DES (Data Encryption Standard) war für kommerzielle Anwendungen hinreichend sicher bis zum Jahr 1982. Daher betrachten wir den Berechnungsaufwand zum Brechen des DES durch Brute-Force-Attacke, also $0.5 \cdot 10^6$ MIPS-Jahre gemäß [LV01], als in der Praxis nicht durchführbar bis zum Jahr 1982.

2. Alle 18 Monate verdoppelt sich die Rechenpower, die man für einen festen Geldbetrag (z.B. 1 US-Dollar) erwerben kann.

$\lceil \log \lvert\Delta\rvert \rceil$	$\#_{\text{Jahre 450MHz, Grad 3}}$	$\#_{\text{MY, Grad 3}}$	$\#_{\text{Jahre 450MHz, Grad 4}}$	$\#_{\text{MY, Grad 4}}$
256	87.8	$3.5 \cdot 10^4$	548.7	$2.2 \cdot 10^5$
348	$4.6 \cdot 10^5$	$1.9 \cdot 10^8$	$2.9 \cdot 10^6$	$1.2 \cdot 10^9$
512	$2.3 \cdot 10^{11}$	$9.2 \cdot 10^{13}$	$1.4 \cdot 10^{12}$	$5.8 \cdot 10^{14}$
640	$1.8 \cdot 10^{15}$	$7.3 \cdot 10^{17}$	$1.1 \cdot 10^{16}$	$4.6 \cdot 10^{18}$
768	$6.7 \cdot 10^{18}$	$2.7 \cdot 10^{21}$	$4.2 \cdot 10^{19}$	$1.7 \cdot 10^{22}$
896	$1.4 \cdot 10^{22}$	$5.5 \cdot 10^{24}$	$8.7 \cdot 10^{22}$	$3.5 \cdot 10^{25}$
1024	$1.8 \cdot 10^{25}$	$7.1 \cdot 10^{27}$	$1.1 \cdot 10^{26}$	$4.4 \cdot 10^{28}$
1280	$9.8 \cdot 10^{30}$	$3.9 \cdot 10^{33}$	$6.1 \cdot 10^{31}$	$2.4 \cdot 10^{34}$
1536	$1.7 \cdot 10^{36}$	$6.9 \cdot 10^{38}$	$1.1 \cdot 10^{37}$	$4.3 \cdot 10^{39}$
1792	$1.3 \cdot 10^{41}$	$5.1 \cdot 10^{43}$	$7.9 \cdot 10^{41}$	$3.1 \cdot 10^{44}$
2048	$4.7 \cdot 10^{45}$	$1.9 \cdot 10^{48}$	$2.9 \cdot 10^{46}$	$1.2 \cdot 10^{49}$
2560	$1.2 \cdot 10^{54}$	$4.9 \cdot 10^{56}$	$7.7 \cdot 10^{54}$	$3.1 \cdot 10^{57}$
3072	$6.1 \cdot 10^{61}$	$2.4 \cdot 10^{64}$	$3.8 \cdot 10^{62}$	$1.5 \cdot 10^{65}$
3584	$8.3 \cdot 10^{68}$	$3.3 \cdot 10^{71}$	$5.2 \cdot 10^{69}$	$2.1 \cdot 10^{72}$
4096	$4.0 \cdot 10^{75}$	$1.6 \cdot 10^{78}$	$2.5 \cdot 10^{76}$	$1.0 \cdot 10^{79}$

Tabelle 4.6: Schlüssellängen und Laufzeit zur Klassenzahlberechnung in Jahren auf einer 450 MHz UltraSparc II-Workstation bzw. Aufwand in MIPS-Jahren

3. Alle 10 Jahre verdoppeln sich die finanziellen Mittel, die Angreifer bereit und in der Lage sind, zum Brechen von Kryptosystemen zu investieren.

4. Es wird keine substanziellen Fortschritte bei der Kryptoanalyse von NF-Kryptosystemen geben.

Die Autoren machen die Hypothesen 1 bis 3 in [LV01] plausibel und illustrieren deren Folgerungen. Sie treffen die Annahme 4 für Signaturverfahren vom Typ DSA und Schnorr (DL-Probleme in Untergruppen mit Primzahlordnung) und für Elliptische Kurven-Verfahren.

Folgt man unseren Annahmen, so berechnen wir zunächst für ein Jahr y den Berechnungsaufwand in MIPS-Jahren, der gemäß [LV01] als praktisch nicht investierbar gilt:

$$\mathrm{IMY}(y) = 0.5 \cdot 10^6 \cdot 2^{2(y-1982)/3} \cdot 2^{(y-1982)/10}$$

Der resultierende Wert IMY(y) wird benutzt, um die Schlüssellängen zu bestimmen, die bis zum Jahr y hinreichende Sicherheit bieten. Für die NF-Kryptosysteme über Zahlkörpern der Grade 3 und 4 berechnen wir hierzu die betragsmäßig kleinste Diskriminante Δ mit $L(|\Delta|)[\frac{1}{2}, \sqrt{2}+o(1)] \geqslant \mathrm{IMY}(y)$, wobei wir den $o(1)$-Term vernachlässigen.

In den resultierenden Tabellen 4.8 und 4.9 findet man sicherheitstechnisch äquivalente Schlüssellängen für RSA, Elliptische Kurven-Kryptosysteme (ECC) und für NF-Kryptosysteme[8] über Zahlkörpern der Grade 3 und 4. Wenn man heute einen Text verschlüsselt, der mindestens bis zum Jahr 2010 geheim bleiben soll, genügt es demnach, einen 1369 Bit-RSA-Key, einen 457 Bit-NF-Kryptosystem-Key bei Verwendung eines Zahlkörpers vom Grad 3, einen 434

[8] in der Tabelle mit NF-Krypto bezeichnet

Jahr	RSA	IQ-DSA	NF-Krypto, Grad 3	NF-Krypto, Grad 4	$\#_{\mathrm{MY}}$
1994	768	540	358	337	$4.3 \cdot 10^8$
2001	1024	687	404	382	$2.0 \cdot 10^{10}$
2014	1536	958	483	459	$9.8 \cdot 10^{12}$
2023	2048	1208	550	525	$1.4 \cdot 10^{15}$
2040	3072	1665	668	641	$4.6 \cdot 10^{18}$
2050	4096	2084	769	740	$2.8 \cdot 10^{21}$

Tabelle 4.7: Vergleich der Schlüssellängen von RSA mit IQ-DSA und RDSA

Bit-NF-Kryptosystem-Key bei Verwendung eines Zahlkörpers vom Grad 4 oder einen 146 Bit-ECC-Key zu verwenden.

Lenstra und Verheul erläutern in ihrer Arbeit [LV01], welche Modifikationen man in den Tabellen vorzunehmen hat, wenn man einzelnen Annahmen, insbesondere Annahme 1, in der oben formulierten Fassung nicht folgen möchte. Diese Ausführungen gelten auch für unseren Kontext.

Tabelle 4.7 listet üblicherweise verwendete Schlüssellängen der Public-Key-Verfahren RSA und IQ-DSA [9] mit den sicherheitstechnisch äquivalenten Schlüssellängen von NF-Kryptosystemen wie RDSA[10] für die Grade 3 und 4 auf. Die Schlüssellänge wird als sehr wichtiger Designparameter moderner Public-Key-Kryptosysteme angesehen. In diesem Sinne sind NF-Kryptosysteme den anderen in der Tabelle dargestellten Public-Key-Kryptosystemen überlegen.

Folgerungen und offene Fragen

Wir haben die Sicherheit von Kryptosystemen über algebraischen Zahlkörpern untersucht, die auf dem Wurzelproblem oder dem diskreten Logarithmusproblem basieren. Für festen Zahlkörpergrad bestimmten wir die Laufzeit des heute effizientesten Algorithmus zum Brechen dieser Kryptosysteme, Buchmanns Klassengruppenalgorithmus:

$$L(|\Delta|)[\frac{1}{2}, \sqrt{2} + o(1)],$$

während man bislang die beste bewiesene obere Schranke der Laufzeit $L(|\Delta|)[\frac{1}{2}, 1.7 + o(1)]$ ([Buc89]) auch für die tatsächliche Laufzeit hielt. Wurzelproblem und diskretes Logarithmusproblem über algebraischen Zahlkörpern vom Grad > 2 scheinen daher noch schwieriger zu sein als die entsprechenden Probleme über imaginär-quadratischen Zahlkörpern (Komplexität $L(|\Delta|)[\frac{1}{2}, 1 + o(1)]$, siehe [HM00]), das diskrete Logarithmusproblem über endlichen Körpern oder das Faktorisierungsproblem ganzer Zahlen (beide haben Komplexität $L(n)\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}} + o(1)\right]$, siehe [LV01]). Durch Messen von Laufzeiten für kleinere Diskriminantenlängen und Extrapolation zeigten wir zum ersten Mal, wie Schlüssellängen für NF-Kryptosysteme[11] zu wählen sind, um das gewünschte Sicherheitslevel zu erreichen. Dabei haben wir das Modell von Lenstra und Verheul

[9] IQ-DSA ist im Wesentlichen die Implementierung von DSA in Klassengruppen imaginär-quadratischer Zahlkörper, siehe [HM00]

[10] in der Tabelle mit NF-Krypto bezeichnet

[11] in Zahlkörpern vom Grad 3 oder 4

Jahr	RSA	NF-Krypto, Grad 3	NF-Krypto, Grad 4	ECC	$\#_{\mathrm{MY}}$
1982	417	284	265	105	$5.0 \cdot 10^5$
1984	463	295	276	108	$1.5 \cdot 10^6$
1986	513	306	287	111	$4.2 \cdot 10^6$
1988	566	318	298	114	$1.2 \cdot 10^7$
1990	622	330	309	117	$3.5 \cdot 10^7$
1991	652	336	315	119	$6.0 \cdot 10^7$
1992	682	342	321	120	$1.0 \cdot 10^8$
1993	713	348	327	121	$1.7 \cdot 10^8$
1994	744	354	333	123	$2.9 \cdot 10^8$
1995	777	361	339	124	$5.0 \cdot 10^8$
1996	810	366	345	126	$8.5 \cdot 10^8$
1997	844	373	351	127	$1.5 \cdot 10^9$
1998	879	379	357	129	$2.5 \cdot 10^9$
1999	915	385	363	130	$4.2 \cdot 10^9$
2000	952	391	369	132	$7.1 \cdot 10^9$
2001	990	398	375	133	$1.2 \cdot 10^{10}$
2002	1028	405	382	135	$2.1 \cdot 10^{10}$
2003	1068	411	388	136	$3.5 \cdot 10^{10}$
2004	1108	417	395	138	$6.0 \cdot 10^{10}$
2005	1149	424	401	139	$1.0 \cdot 10^{11}$
2006	1191	430	408	141	$1.7 \cdot 10^{11}$
2007	1235	437	414	142	$2.9 \cdot 10^{11}$
2008	1279	444	421	144	$5.0 \cdot 10^{11}$
2009	1323	451	427	145	$8.5 \cdot 10^{11}$
2010	1369	457	434	146	$1.5 \cdot 10^{12}$
2011	1416	464	441	148	$2.5 \cdot 10^{12}$
2012	1464	471	448	149	$4.2 \cdot 10^{12}$
2013	1513	478	454	151	$7.1 \cdot 10^{12}$
2014	1562	485	461	152	$1.2 \cdot 10^{13}$
2015	1613	492	468	154	$2.1 \cdot 10^{13}$
2016	1664	499	475	155	$3.5 \cdot 10^{13}$
2017	1717	506	482	157	$6.0 \cdot 10^{13}$
2018	1771	514	489	158	$1.0 \cdot 10^{14}$
2019	1825	521	496	160	$1.7 \cdot 10^{14}$

Tabelle 4.8: Schlüssellängen verschiedener Signaturverfahren für gleiches Sicherheitsniveau

Jahr	RSA	NF-Krypto, Grad 3	NF-Krypto, Grad 4	ECC	$\#_{\mathrm{MY}}$
2020	1881	528	503	161	$2.9 \cdot 10^{14}$
2021	1937	535	510	163	$5.0 \cdot 10^{14}$
2022	1995	543	518	164	$8.5 \cdot 10^{14}$
2023	2054	550	525	166	$1.5 \cdot 10^{15}$
2024	2113	558	532	167	$2.5 \cdot 10^{15}$
2025	2174	565	539	169	$4.2 \cdot 10^{15}$
2026	2236	573	547	170	$7.1 \cdot 10^{15}$
2027	2299	580	554	172	$1.2 \cdot 10^{16}$
2028	2362	588	562	173	$2.1 \cdot 10^{16}$
2029	2427	595	569	175	$3.5 \cdot 10^{16}$
2030	2493	603	577	176	$6.0 \cdot 10^{16}$
2032	2629	619	592	179	$1.7 \cdot 10^{17}$
2034	2768	634	607	182	$5.0 \cdot 10^{17}$
2036	2912	650	623	185	$1.5 \cdot 10^{18}$
2038	3061	666	639	188	$4.2 \cdot 10^{18}$
2040	3214	683	655	191	$1.2 \cdot 10^{19}$
2042	3371	699	671	194	$3.5 \cdot 10^{19}$
2044	3533	716	687	197	$1.0 \cdot 10^{20}$
2046	3700	732	704	200	$3.0 \cdot 10^{20}$
2048	3871	749	720	203	$8.5 \cdot 10^{20}$
2050	4047	766	737	206	$2.5 \cdot 10^{21}$
2056	4608	819	788	289	$6.0 \cdot 10^{22}$
2061	5120	863	832	303	$8.5 \cdot 10^{23}$
2070	6144	947	914	328	$1.0 \cdot 10^{26}$
2079	7168	1033	1000	354	$1.2 \cdot 10^{28}$
2086	8192	1103	1068	374	$4.9 \cdot 10^{29}$
2094	9216	1185	1149	396	$3.4 \cdot 10^{31}$
2100	10240	1248	1212	413	$8.2 \cdot 10^{32}$
2107	11264	1324	1286	433	$3.4 \cdot 10^{34}$
2113	12288	1390	1352	450	$8.1 \cdot 10^{35}$
2118	13312	1447	1408	464	$1.2 \cdot 10^{37}$
2124	14336	1516	1476	481	$2.8 \cdot 10^{38}$
2129	15360	1574	1534	495	$4.0 \cdot 10^{39}$
2134	16384	1634	1593	509	$5.6 \cdot 10^{40}$

Tabelle 4.9: Schlüssellängen verschiedener Signaturverfahren für gleiches Sicherheitsniveau (Forts.)

([LV01]) angewendet, in dem die Schlüssellängen der NF-Kryptosysteme sicherheitstechnisch mit Schlüssellängen traditioneller Public-Key-Verfahren in Bezug gesetzt werden.

Offen bleibt die Frage, wie man die passenden Schlüssellängen der NF-Kryptosysteme für größeren Zahlkörpergrad als 4 bestimmt. Man ist hier mit dem praktischen Problem konfrontiert, dass man nicht genügend viele Zeitmessungen vornehmen kann, da Buchmanns Algorithmus bereits für sehr kleine Eingabeparamter sehr ineffizient ist. Bislang vermutet man, dass die Laufzeit des Buchmann-Algorithmus exponentiell im Zahlkörpergrad wächst ([BP97]).

Kapitel 5

Effiziente NF-Arithmetik

Die von üblichen Computeralgebrasystemen wie PARI/GP, KANT/KASH und LiDIA zur Verfügung gestellten Datenstrukturen und Algorithmen für Ideale algebraischer Zahlkörper sind nicht hinreichend effizient für den praktischen kryptographischen Gebrauch. LiDIA ist diesbezüglich wenigstens so effizient wie die Systeme PARI/GP und KANT/KASH (siehe [Nei02]). Unsere erste Implementierung des Signaturverfahrens RDSA in Stender-Körpern benutzte die von LiDIA zur Verfügung gestellten Datenstrukturen und Algorithmen. Eine RDSA-Signatur in einem kubischen Stender-Körper mit 501-Bit-Diskriminante benötigte auf einer UltraSPARC-III-Maschine mit 333 MHz etwa 91 Sekunden ([HMNP99]). Dies war einerseits der grundlegende Nachweis, dass man überhaupt kryptographische Verfahren in algebraischen Zahlkörpern implementieren kann. Andererseits wurde jedoch deutlich, dass eine effizientere Arithmetik für Zahlkörper und ihre Objekte erfunden und implementiert werden musste sowie ggf. weitere Effizienzoptimierungen bei der Implementierung der kryptographischen Verfahren erzielt werden mussten.

In diesem Kapitel stellen wir eine effiziente Arithmetik für algebraische Zahlkörper vor. In Abschnitt 5.1 beschreiben wir zunächst, wie man gemäß unseres Vorschlages die mathematischen Objekte im Rechner darstellen soll. Im anschließenden Abschnitt 5.2 formulieren wir die Algorithmen für diese Objekte, welche man zur Implementierung der kryptographischen Verfahren aus Kapitel 3 benötigt. Viele der von uns vorgeschlagenen Algorithmen sind deutlich zeiteffizienter als die traditionellen Algorithmen. In Abschnitt 5.3 beschreiben wir Details unserer Implementierungen für kryptographische Protokolle über algebraischen Zahlkörpern. Wir präsentieren die Laufzeiten unserer Verfahren und vergleichen diese mit den Laufzeiten von RSA als Stellvertreter traditioneller Verfahren.

Die Ausführungen in diesem Kapitel lehnen sich teilweise an unserer Veröffentlichung [MNP01] an.

5.1 Darstellung der mathematischen Objekte

5.1.1 Darstellung von Zahlkörpern, Ordnungen und Klassengruppen

Wir müssen algebraische Zahlkörper und Ordnungen im Computer darstellen. Algebraische Zahlkörper stellen wir durch ihr irreduzibles monisches erzeugendes Polynom f dar. Die Darstellung im Computer ist das n-Tupel der Koeffizienten (ganze Zahlen) des Polynoms, wobei

n der Zahlkörpergrad ist. (Den höchsten Koeffizienten des Polynoms speichern wir nicht, da er immer 1 ist.) Berechnungen im Zahlkörper können dann mittels Polynom-Arithmetik modulo f durchgeführt werden. Daher können wir eine algebraische Zahl α durch ein Polynom modulo f im Computer repräsentieren, d.h. wir haben $\alpha = \frac{1}{d}\sum_{i=0}^{n-1} a_i x^i$ mit $d, a_i \in \mathbb{Z}$. Durch Speicherung des Polynoms f definieren wir implizit die Ordnung $\mathbb{Z}[x]/(f)$, die aus den algebraischen Zahlen mit Nenner $d = 1$ besteht. Wir wollen in der Regel allerdings in einer anderen Ordnung, der Maximalordnung, rechnen. Die Maximalordnung kann nicht immer auf die genannte Art repräsentiert werden. Daher speichern wir für Ordnungen zusätzlich eine Transformationsmatrix $T \in \mathbb{Z}^{n\times n}$ und einen Nenner d, um die Ordnung mit $\mathbb{Z}$-Basis $(\omega_0, \ldots, \omega_{n-1}) = (1, x, ..., x^{n-1}) \cdot \frac{1}{d}T$ zu repräsentieren, wobei $\omega_i \in \mathbb{Z}[x]/(f)$. Für Ordnungen mit $T \neq I_n$ speichern wir zusätzlich eine Multiplikationstabelle; diese beschreibt, wie zwei Elemente der Basis multipliziert werden, d.h. wir speichern $w_{i,j,k}, 0 \leqslant i, j, k < n$, mit $\omega_i \cdot \omega_j = \sum_{k=0}^{n-1} w_{i,j,k}\omega_k$. Eine solche Multiplikationstabelle definiert bereits vollständig die Ordnung oder den Zahlkörper. Daher empfehlen wir für Zahlkörper oder Ordnungen im Falle $T \neq I_n$, auf die oben geschilderte Darstellung durch Speicherung eines Polynoms, Nenners und einer Transformationsmatrix zu verzichten. Zu beliebiger gegebener Ordnung eines Zahlkörpers ist in eindeutiger Weise die Klassengruppe der Ordnung definiert. Statt eine Klassengruppe im Rechner darzustellen, stellen wir ihre Ordnung im Rechner dar. Um also die Klassengruppe eines algeraischen Zahlkörpers im Rechner darzustellen, stellen wir die Maximalordnung des Zahlkörpers dar.

Erzeugen der Objekte Zahlkörper, Ordnung und Klassengruppe
Das Objekt Zahlkörper erzeugen wir im Rechner durch Wahl und Fixierung des erzeugenden irreduziblen monischen Polynoms f vom Zahlkörpergrad n. Hiervon ausgehend erzeugen wir das Objekt Ordnung im Computer, indem wir zusätzlich eine unimodulare Matrix $T \in \mathbb{Z}^{n\times n}$ und einen Nenner $d \in \mathbb{Z}_{>0}$ fixieren. (Anschließend berechnen wir ggf. die Multiplikationstabelle zu der entsprechenden Ordnung und löschen f, d und T.) Die Klassengruppe der entsprechenden Ordnung ist somit ebenfalls im Rechner dargestellt.

5.1.2 Darstellung von algebraischen Zahlen

Wir speichern algebraische Zahlen als Koeffizientenvektoren mit Referenz auf eine $\mathbb{Z}$-Basis und einen Nenner, wobei alle Koeffizienten und der Nenner ganze Zahlen sind. Wir repräsentieren $\alpha = \frac{1}{d}\sum_{i=0}^{n-1} a_i\omega_i$ im Computer also als n-Tupel $(d, (a_0, \ldots, a_{n-1}))$. Addition und Subtraktion können dann komponentenweise durchgeführt werden, sobald wir einen gemeinsamen Nenner für die beiden involvierten algebraischen Zahlen gefunden haben. Algebraische Zahlen können durch Benutzung von Polynomarithmetik oder durch Verwenden der Multiplikationstabelle (siehe oben) multipliziert und dividiert werden.

5.1.3 Darstellung von Idealen.

Darstellung in anderen Computeralgebrasystemen Andere Computeralgebrasysteme repräsentieren ein (gebrochenes) Ideal $\mathfrak{a}$ durch einen Nenner $d \in \mathbb{Z}$ und eine $n \times n$-Matrix $M_\mathfrak{a}$ in Hermite-Normalform, welche eine $\mathbb{Z}$-Basis $(\alpha_1, \ldots, \alpha_n)$ des Ideals $d\mathfrak{a}$ repräsentiert. Die Einträge der Matrix $M_\mathfrak{a}$ sind ganze Zahlen und dem Betrage nach kleiner oder gleich $|\Delta|$. Die Spalten von $M_\mathfrak{a}$ sind die Koeffizientenvektoren zur Darstellung der n algebraischen Zahlen $\alpha_1, \ldots, \alpha_n \in \mathcal{O}$

bzgl. einer fixierten Ordnung mit $\mathbb{Z}$-Basis $(\omega_1, \ldots, \omega_n)$:

$$\begin{aligned} \mathfrak{a} &= \alpha_1\mathbb{Z} + \ldots + \alpha_n\mathbb{Z} \qquad (\alpha_i \in \mathcal{O}) \\ &= ((\omega_1, \ldots, \omega_n) M_\mathfrak{a})\, \mathbb{Z} \\ & \qquad M_\mathfrak{a} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \in \mathbb{Z}^{n\times n} \end{aligned}$$

Diese Darstellung von Idealen nennen wir im Folgenden $\mathbb{Z}$-*Basis-Darstellung*.

Unsere Darstellungen Je nach Kontext wählen wir eine der beiden folgenden zeit- und platzeffizienteren Varianten: die LiDIA-Darstellung (Abschnitt 5.1.3.1) oder die Zwei-Element-Darstellung von Idealen (Abschnitt 5.1.3.2).

5.1.3.1 LiDIA-Darstellung von Idealen

Zur Darstellung des (gebrochenen) Ideals $\mathfrak{a} = \alpha_1\mathbb{Z} + \ldots + \alpha_n\mathbb{Z}$ berechnen wir bei gegebener $\mathbb{Z}$-Basis-Darstellung mit Darstellungsmatrix $M_\mathfrak{a} = (a_{ij})_{i,j}$ eine positive Zahl $m \in \mathbb{Z}$ mit $m\mathcal{O} \subset \mathfrak{a}$. (Die kleinste Zahl mit dieser Eigenschaft heißt *Exponent* von $\mathfrak{a}$.) Wegen

$$\mathfrak{a} + m\mathcal{O} \subseteq \mathfrak{a} + \mathfrak{a} \subseteq \mathfrak{a}$$
$$\text{gilt } \mathfrak{a} = \alpha_1\mathbb{Z} + \ldots + \alpha_n\mathbb{Z} + m\mathcal{O}$$

Daher hat $\mathfrak{a}$ die Darstellungsmatrix

$$M'_\mathcal{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} & m & & 0 \\ \vdots & & \vdots & & \ddots & \\ a_{n1} & \cdots & a_{nn} & 0 & & m \end{pmatrix} \in \mathbb{Z}^{n\times 2n}$$

$$\rightsquigarrow \quad \begin{pmatrix} a_{11} \bmod m & \cdots & a_{1n} \bmod m & m & & 0 \\ \vdots & & \vdots & & \ddots & \\ a_{n1} \bmod m & \cdots & a_{nn} \bmod m & 0 & & m \end{pmatrix} \in \mathbb{Z}^{n\times 2n}$$

$$\rightsquigarrow \quad \begin{pmatrix} \tilde{a}_{11} \bmod m & \cdots & \tilde{a}_{1r} \bmod m & m & & 0 \\ \vdots & & \vdots & & \ddots & \\ \underbrace{\tilde{a}_{n1} \bmod m \quad \cdots \quad \tilde{a}_{nr} \bmod m}_{\bar{A}} & & & 0 & & m \end{pmatrix} \in \mathbb{Z}^{n\times(n+r)} \quad (1 \leqslant r \leqslant n)$$

Wir repräsentieren das gebrochene Ideal durch $(m, \bar{A})$. Die Matrix $\bar{A}$ mit Einträgen in $\mathbb{Z}/m\mathbb{Z}$ entsteht also aus der Darstellungsmatrix $M_\mathfrak{a}$ durch komponentenweise Reduktion modulo m. Wir speichern $\bar{A}$ als reduzierte Matrix mit so wenig Spalten wie möglich (Details siehe [Nei98, Nei02]). Diese Darstellung ist nicht eindeutig. Die geschilderte Darstellung von Idealen nennen wir im Folgenden *LiDIA-Darstellung* ([LiD01]). Experimentelle Untersuchungen zeigen, dass die LiDIA-Darstellung in der Praxis platzeffizienter als die $\mathbb{Z}$-Basis-Darstellung ist ([Nei02]). Um zu überprüfen, ob zwei Ideale gleich sind, bestimmen und vergleichen wir die Exponenten beider Ideale und berechnen und vergleichen jeweils die eindeutige Darstellung des Moduls, der von den

Spalten von $\bar{A}$ erzeugt wird (Details siehe [How86]). Die LiDIA-Darstellung ist unsere Standard-Darstellung von Idealen. Die im Folgenden beschriebene Zwei-Element-Darstellung verwenden wir nur für Primideale (eigener Unterabschnitt, siehe unten) und für Ideale, von denen wir wissen, dass wir später ein Ideal in LiDIA-Darstellung damit multiplizieren müssen. Beispielsweise berechnen wir für eine Exponentiation $[\mathfrak{a}]^k$ vorab LLL-reduzierte Vertreter der 2-Potenzen $[\mathfrak{a}]^{2^1}, [\mathfrak{a}]^{2^2}, \ldots$ in Zwei-Element-Darstellung.

5.1.3.2 Zwei-Element-Darstellung von Idealen

Jedes gebrochene Ideal $\mathfrak{a}$ der Maximalordnung $\mathcal{O}$ von K besitzt eine Darstellung der Form

$$\mathfrak{a} = a\mathcal{O} + \alpha\mathcal{O} \ ,$$

wobei $a \in \mathbb{Z}_{>0} \cap \mathfrak{a}$ und $\alpha \in \mathfrak{a}$ ([PZ89]). Eine solche Darstellung nennen wir *Zwei-Element-Darstellung* des Ideals. In der Zwei-Element Darstellung benötigen wir nur $n+1$ ganze Zahlen, um ein Ideal im Computer darzustellen, während man in der $\mathbb{Z}$-Basis-Darstellung n^2 ganze Zahlen und in der LiDIA-Darstellung $n \cdot r \leqslant n^2$ ganze Zahlen ($1 \leqslant r \leqslant n$) benötigt.

Berechnung von Zwei-Element-Darstellungen Zu gegebenem Ideal $\mathfrak{a}$ in LiDIA-Darstellung müssen wir in der Praxis häufig eine Zwei-Element-Darstellung berechnen. Hierzu dient Algorithmus 5.1.

Algorithmus 5.1 Berechnung einer Zwei-Element-Darstellung eines Ideals

Eingabe: Ein Ideal $\mathfrak{a}$ in LiDIA-Darstellung.
Ausgabe: $\{a, \alpha\}$ mit $\mathfrak{a} = a\mathcal{O} + \alpha\mathcal{O}$ oder Error.

1: Berechne $\mathbb{Z}$-Basis-Darstellung von $\mathfrak{a}$: $\mathfrak{a} = \alpha_1\mathbb{Z} + \ldots + \alpha_n\mathbb{Z}$
2: Berechne $a := N(\mathfrak{a})$.
3: **for** $\alpha \in \left\{ \sum_{i=0}^{n-1} e_i\alpha_i : \ e_i \in \{-1, 0, 1\} \right\}$ **do**
4: **if** $N(a\mathcal{O} + \alpha\mathcal{O}) = N(\mathfrak{a})$ **then return** $\{a, \alpha\}$
5: **end for**
6: **return** Error

Die Vorgehensweise im Algorithmus ist die folgende:
Nach Berechnung einer $\mathbb{Z}$-Basis-Darstellung $\mathfrak{a} = \alpha_1\mathbb{Z} + \ldots + \alpha_n\mathbb{Z}$ wird die positive ganze Zahl $a = N(\mathfrak{a})$ als erster Erzeuger gewählt. Anschließend wird ein zweiter Erzeuger α geschickt geraten: Für alle $\alpha \in \left\{ \sum_{i=0}^{n-1} e_i\alpha_i : \ e_i \in \{-1, 0, 1\} \right\}$ wird getestet, ob $a\mathcal{O} + \alpha\mathcal{O} = \mathfrak{a}$. In diesem Fall kann der Algorithmus mit der Ausgabe zweier Erzeuger abgebrochen werden. Wegen $a\mathcal{O} + \alpha\mathcal{O} \subseteq \mathfrak{a}$ genügt es hier zu prüfen, ob $N(a\mathcal{O} + \alpha\mathcal{O}) = N(\mathfrak{a})$.

Der Algorithmus ist zwar probabilistisch, terminiert in unseren Experimenten aber bei zufällig gewählten Idealen $\mathfrak{a}$ fast immer (Wahrscheinlichkeit $> 99.9\%$) mit Ausgabe einer Zwei-Element-Darstellung. Die Zwei-Element-Darstellung von Idealen verwenden wir insbesondere zur Speicherung vorberechneter 2-Potenzen bei der Exponentiation einer Idealklasse.

5.1.4 Darstellung von Primidealen.

Wenn wir in unseren NF-Kryptoverfahren mit Primidealen rechnen, so müssen wir in der Regel Potenzprodukte einer großen Anzahl von Primidealen berechnen. Hierfür wählen wir die *Zwei-Element-Darstellung*, die sich als besonders platz- und zeiteffizient erweist. Wir stellen ein Primideal $\mathfrak{p}$ dar durch ein Paar (p, α), wobei $p \in \mathbb{Z}$ prim und α eine algebraische Zahl in unserer Ordnung $\mathcal{O}$ ist. Die Ordnung $\mathcal{O}$ merken wir uns implizit über α. Es ist $\mathfrak{p} = p\mathcal{O} + \alpha\mathcal{O}$.

5.1.5 Darstellung von Idealklassen.

Wir stellen Idealklassen durch ein beliebiges LLL-reduziertes Ideal $\mathfrak{a}$ der Klasse (siehe unten) dar. Eine Idealklasse wird also im Computer repräsentiert durch ein Paar $(e, \bar{A})$, wobei e eine positive ganze Zahl ist mit $e\mathcal{O} \subset \mathfrak{a}$. (In der Regel wählen wir die kleinste positive Zahl (den Exponenten) mit dieser Eigenschaft, und es gilt $1 \leqslant e \leqslant |\Delta|$.) Die Matrix $\bar{A}$ ist eine $n \times r$-Matrix $(1 \leqslant r \leqslant n)$ mit Einträgen in $\mathbb{Z}/e\mathbb{Z}$. Die LLL-reduzierten Ideale nehmen für Idealklassen etwa die Rolle der kleinsten positiven Reste von Restklassen ein, also bei Berechnungen modulo einer ganzen Zahl. Im Gegensatz zum kleinsten positiven Vertreter einer Restklasse ist der LLL-reduzierte Idealvertreter allerdings nicht eindeutig bestimmt, denn es gibt viele LLL-reduzierte Vertreter einer Idealklasse.

5.2 Effiziente Algorithmen

Sei K ein algebraischer Zahlkörper vom Grad n mit Maximalordnung $\mathcal{O}$, Diskriminante Δ, Klassengruppe $Cl(K)$ und Klassenzahl $h(K)$. Seien $\mathfrak{a}, \mathfrak{b}$ zwei Ideale von $\mathcal{O}$. (Die folgenden Algorithmen lassen sich auch auf Nichtmaximalordnungen anwenden.)

5.2.1 Bestimmen der Maximalordnung

Im Allgemeinen müssen wir in den NF-Kryptoverfahren in der Maximalordnung rechnen. Im Falle der kubischen Stender-Körper ist die Maximalordnung explizit bekannt (siehe Abschnitt 4.3). In allen anderen Fällen wenden wir zur Bestimmung der Maximalordnung eines Zahlkörpers den Round 2-Algorithmus an (siehe [Coh95]).

5.2.2 Die Gruppenoperation

Wie in Unterabschnitt 5.1 geschildert repräsentieren wir Gruppenelemente (Idealklassen) durch ein reduziertes Ideal der Idealklasse. Wir realisieren die Gruppenoperation $[\mathfrak{a}] \cdot [\mathfrak{b}]$ durch Multiplikation der beiden repräsentierenden Ideale $\mathfrak{a}, \mathfrak{b}$ und LLL-Reduktion des Idealproduktes $\mathfrak{ab}$.

5.2.2.1 Reduktion eines Ideals.

Folgendermaßen reduzieren wir ein Ideal $\mathfrak{a}$: Wir berechnen Minkowskis Einbettung, d.h. für die Elemente einer $\mathbb{Z}$-Basis des Ideals berechnen wir die (vereinfachten) Konjugiertenvektoren. So erhalten wir ein Gitter im $\mathbb{R}^n$. Mit dem LLL-Algorithmus berechnen wir einen kurzen Vektor dieses Gitters. Diesen kurzen Gittervektor interpretieren wir als LLL-reduziertes Ideal in der Idealklasse $[\mathfrak{a}]$.

5.2.2.2 Idealmultiplikation.

Zur Idealmultiplikation benutzen wir in Abhängigkeit vom Kontext unterschiedliche Algorithmen. Diese schildern wir im Folgenden.

Mit den Methoden zur Multiplikation von Idealen und der LLL-Reduktion des Idealproduktes können wir die gängigen generischen Algorithmen zur effizienten Exponentiation in Klassengruppen anwenden (siehe [MOV97]).

Potenzierung von Primidealen. Potenzen von $\mathfrak{p} = p\mathcal{O} + \alpha\mathcal{O}$ können wir leicht mittels der folgenden Gleichung bestimmen: $\mathfrak{p}^k = p^{\lceil k/z \rceil}\mathcal{O} + \alpha^k\mathcal{O}$. Hier bezeichnet z den Ramifizierungsindex von p in $\mathfrak{p}$ (siehe [Coh95]).

Ideal · Primideal. Wir multiplizieren ein Ideal $\mathfrak{a}$ in LiDIA-Darstelung mit einem Primideal $\mathfrak{p} = p\mathcal{O} + \alpha\mathcal{O}$, indem wir die Ideale $p\mathfrak{a}$ und $\alpha\mathfrak{a}$ addieren.

Ideal · Ideal. Zur Multiplikation von Idealen, die kein Primideal sind, verwenden wir in Abhängigkeit vom Kontext unterschiedliche Algorithmen.

Ideal in LiDIA-Darstellung · Ideal in LiDIA-Darstellung
Algorithmus 5.2 zeigt auf, wie wir zwei Ideale in LiDIA-Darstellung miteinander multiplizieren. Der Korrektheitsbeweis für Algorithmus 5.2 findet sich in [Nei02, S.69].

Algorithmus 5.2 Multiplikation zweier Ideale in LiDIA-Darstellung

Eingabe: Zwei ganze Ideale $\mathfrak{a} = (\ell, \bar{A}) = (\ell, (\underline{a_1}, \ldots, \underline{a_{r_1}}))$ und $\mathfrak{b} = (m, \bar{B}) = (m, (\underline{b_1}, \ldots, \underline{b_{r_2}}))$ in LiDIA-Darstellung.
Ausgabe: Ein ganzes Ideal $\mathfrak{c}$ in LiDIA-Darstellung mit $\mathfrak{c} = \mathfrak{a}\mathfrak{b}$.

1: Berechne $\mathfrak{c} := (\ell \cdot m, (\underline{a_1 b_1}, \ldots, \underline{a_1 b_{r_2}}, \ldots, \underline{a_{r_1} b_1}, \ldots, \underline{a_{r_1} b_{r_2}}, m\underline{a_1}, \ldots, m\underline{a_{r_1}}, \ell\underline{b_1}, \ldots, \ell\underline{b_{r_2}}))$.
2: **return** $[\mathfrak{c}]$

Ideal in LiDIA-Darstellung · Ideal in Zwei-Element-Darstellung
Seien ein Ideal $\mathfrak{a}$ in LiDIA-Darstellung $(e, \bar{A})$ und ein Ideal $\mathfrak{b} = b\mathcal{O} + \beta\mathcal{O}$ in Zwei-Element-Darstellung gegeben. Dann berechnen wir das Idealprodukt $\mathfrak{a}\mathfrak{b}$ in LiDIA-Darstellung folgendermaßen:

$$\mathfrak{a}\mathfrak{b} = \mathfrak{a} \cdot b + \mathfrak{a} \cdot \beta\,.$$

Eine detailliertere Beschreibung unserer Vorgehensweise bietet Algorithmus 5.3. Zur Hermite-Normalformberechnung (HNF) verwenden wir die optimierten Versionen aus [The00]. Einfache Algorithmen zur HNF-Berechnung werden beispielsweise in [Coh95] beschrieben. Für eine technisch noch exaktere Beschreibung zu der von uns verwendeten Multiplikation ganz-algebraischer Zahlen mit Idealen und zur Addition von Idealen verweisen wir auf [Nei98, Nei02].

$\lceil \log \lvert\Delta\rvert \rceil$	**D**	**LiDIA $*$ LiDIA**	**LiDIA $*$ Zwei-Elt**	**Faktor**
358	$\sim 5.26 \cdot 10^{17}$	1.49 ms	0.37 ms	4.03
404	$\sim 1.07 \cdot 10^{20}$	1.79 ms	0.42 ms	4.25
550	$\sim 2.27 \cdot 10^{27}$	2.92 ms	0.63 ms	4.67
668	$\sim 1.89 \cdot 10^{33}$	4.02 ms	0.86 ms	4.70
769	$\sim 2.21 \cdot 10^{38}$	5.27 ms	1.13 ms	4.65

Tabelle 5.1: Laufzeiten der Idealmultiplikation in Stender-Körpern vom Grad 3

Algorithmus 5.3 Multiplikation Ideal in LiDIA-Darstellung mit Ideal in Zwei-Element-Darstellung

Eingabe: Ideal $\mathfrak{a} = (e, \bar{A})$ in LiDIA-Darstellung und $\mathfrak{b} = b\mathcal{O} + \beta\mathcal{O}$ in Zwei-Element-Darstellung.

Ausgabe: Ein Ideal $\mathfrak{c}$ in LiDIA-Darstellung mit $\mathfrak{c} = \mathfrak{a}\mathfrak{b}$.

1: // Berechne $\mathfrak{a} \cdot b$:
2: Multipliziere alle Einträge von $\bar{A}$ und den Modul e mit der ganzen Zahl b.
3: // Berechne $\mathfrak{a} \cdot \beta$:
4: Multipliziere die ganz-algebraische Zahl β mit allen Spalten des $\mathbb{Z}$-Erzeugendensystems $(\underbrace{\bar{A}}_{r} \mid \underbrace{e \cdot I_n}_{n})$ von $\mathfrak{a}$.
5: Berechne die HNF der resultierenden Matrix mit $n+r$ Spalten (führt zu einer Matrix mit höchstens n Spalten).
6: // Berechne $\mathfrak{a} \cdot b + \mathfrak{a} \cdot \beta$:
7: Konkateniere die Spalten der Darstellungen von $\mathfrak{a} \cdot b$ und $\mathfrak{a} \cdot \beta$.
8: Berechne die HNF der resultierenden Matrix aus höchstens $n + r$ Spalten (führt zu einer Matrix mit höchstens n Spalten).
9: Interpretiere das Ergebnis als LiDIA-Darstellung eines Ideals $\mathfrak{c}$ und **return** [$\mathfrak{c}$]

Die Tabellen 5.1, 5.2, 5.3 vergleichen die Laufzeiten für die Multiplikation zweier Ideale in LiDIA-Darstellung mit den Laufzeiten für die Multiplikation der gleichen Ideale, wobei eins in LiDIA-Darstellung und das zweite Ideal in Zwei-Element-Darstellung gegeben ist. In den Experimenten verwenden wir LLL-reduzierte Ideale zufällig gewählter Idealklassen in Klassengruppen von Stender-Körpern der Grade $n = 3, 4$ und 6:

$$K = \mathbb{Q}(\sqrt[n]{D^n + 1}) \qquad D \in \mathbb{Z}_{>0}$$

Die erste Spalte gibt Aufschluss über die binäre Länge der Diskriminante Δ, die zweite Spalte über die Größe(nordnung) des Parameters D; die letzte Spalte zeigt den Beschleunigungsfaktor unserer neuen Methode zur Multiplikation von Idealen („LiDIA-Darst. $*$ Zwei-Elt.-Darst.“, vierte Spalte) verglichen mit der in LiDIA implementierten Methode („LiDIA-Darst. $*$ LiDIA-Darst.“, dritte Spalte). Alle Laufzeiten sind in Millisekunden angegeben und über 100 Iterationen gemittelt. Die Tabelle 5.8 gibt Aufschluss über die Systemumgebung, in der alle in diesem Kapitel genannten Laufzeiten gemessen wurden. Die Laufzeiten zeigen, dass unsere neue Methode zur Idealmultiplikation im Vergleich mit der in LiDIA implementierten Methode zu einem Beschleunigungsfaktor von $4.5, 8$ bzw. 14 für die Zahlkörpergrade

$\lceil \log \lvert\Delta\rvert \rceil$	D	LiDIA $*$ LiDIA	LiDIA $*$ Zwei-Elt	Faktor
335	$\sim 1.69 \cdot 10^8$	4.11 ms	0.56 ms	7.39
379	$\sim 2.03 \cdot 10^9$	4.76 ms	0.63 ms	7.52
521	$\sim 1.11 \cdot 10^{13}$	7.63 ms	0.90 ms	8.51
636	$\sim 5.06 \cdot 10^{15}$	10.57 ms	1.28 ms	8.27
741	$\sim 2.95 \cdot 10^{18}$	13.81 ms	1.66 ms	8.32

Tabelle 5.2: Laufzeiten der Idealmultiplikation in Stender-Körpern vom Grad 4

$\lceil \log \lvert\Delta\rvert \rceil$	D	LiDIA $*$ LiDIA	LiDIA $*$ Zwei-Elt	Faktor
336	1880	14.18 ms	1.05 ms	13.48
379	5090	16.56 ms	1.19 ms	13.89
528	159465	26.39 ms	1.87 ms	14.13
634	1846389	36.35 ms	2.54 ms	14.31
738	36392991094	47.92 ms	3.39 ms	14.12

Tabelle 5.3: Laufzeiten der Idealmultiplikation in Stender-Körpern vom Grad 6

$3, 4$ bzw. 6 führt. Allerdings sind Ideale häufig a priori nicht in Zwei-Element-Darstellung, sondern in LiDIA-Darstellung gegeben (z.B. nach der Wahl eines Vertreters einer zufälligen Idealklasse oder nach LLL-Reduktion eines Ideals). Unsere Experimente zeigen: Wenn zwei Ideale nur in LiDIA-Darstellung vorliegen, ist die Berechnung einer Zwei-Element-Darstellung für ein Ideal mit anschließender Anwendung unserer neuen Multiplikationsmethode („LiDIA-Darst. $*$ Zwei-Elt.-Darst.“) ineffizienter als die sofortige Multiplikation der beiden Ideale in LiDIA-Darstellung.

Dennoch können wir unsere neue Multiplikationsmethode gewinnbringend in Situationen einsetzen, in denen man viel Zeit für Vorberechnungen hat (in deren Rahmen wir die Zwei-Element-Darstellungen für später zu verwendende Ideale berechnen) und wenig Zeit für anschließende Operationen (Idealmultiplikation). Bei der Implementierung unserer NF-Kryptosysteme berechnen wir beispielsweise parallel zur Schlüsselerzeugung LLL-reduzierte Vertreter der 16-Potenzen $[\mathfrak{g}]^{16^1}, [\mathfrak{g}]^{16^2}, \ldots$ in Zwei-Element-Darstellung. Die späteren zeitkrischen Exponentiationen $[\mathfrak{g}]^k$ (z.B. bei der Erstellung oder Verifikation einer Signatur) werden dann bei Benutzung unserer neuen Multiplikationsmethode deutlich effizienter.

Die in anderen Computeralgebrasystemen (PARI [BBB+02] und KANT [Poh02]) verwendete klassische Idealmultiplikation unter Verwendung von $\mathbb{Z}$-Basis-Darstellungen der Ideale ist ineffizienter als unsere Methode zur Multiplikation von zwei Idealen in LiDIA-Darstellung (siehe [Nei02] zum Vergleich von Laufzeiten).

Implementierung der neuen Idealmultiplikation – Erweiterung von LiDIA

Primideale werden in LiDIA durch eine eigene Klasse `prime_ideal` dargestellt. LiDIA repäsentiert alle anderen Ideale als Instanzen der Klasse `alg_ideal`. Wegen der engen mathematischen

Verwandtschaft zwischen Idealen und Ordnungen (Ideale sind spezielle Ordnungen) ist die Klasse `alg_ideal` von der Klasse `order` abgeleitet; letztere repräsentiert Ordnungen. Die Datenelemente der Klasse `alg_ideal` sind dabei genau die Datenstrukturen, die wir in Abschnitt 5.1.3 als LiDIA-Darstellung von Idealen vorgestellt hatten.

```
class module{
protected:
#ifdef LIDIA_DEBUG
  static int count;
#endif
#ifndef LIDIA_NO_MUTABLE
  mutable
#endif
  bigmod_matrix base;
  bigint den;
  nf_base * O;
#ifndef LIDIA_NO_MUTABLE
  mutable
#endif
  bool is_exp;

  void compare(const module&, bool &, bool &) const;
    // internal routine for comparisons

public:
  // Contructors & destructor:
  ...
}

class alg_ideal: public module{
public:
  // member functions
  ...
}
```

Zur effizienten Implementierung der Zwei-Element-Darstellung erweiterten wir die LiDIA-Klasse `alg_ideal` folgendermaßen:

1. Neue Datenelemente für Klasse `alg_ideal`

```
bigint gen1;          // first generator of the ideal
alg_number gen2;      // second generator of the ideal
int max_2exp;         // 2^max_2exp is the maximal exponent for which
                      // the power is stored at the moment
                      // 2-powers are stored whenever max_2exp > 0
bigint* p_gen1;       // pointer to first generators of the
```

```
                        // 2-powers of the ideal class
  alg_number* p_gen2;   // pointer to second generators of the
                        // 2-powers of the ideal class
```

2. Neue Elementfunktionen für Klasse `alg_ideal`

```
module & operator =(const module & A)
  {  multiply(*this, A, alg_ideal_cast(order(A.which_base())));
     this->max_2exp = -1;
     return *this; }
    //original code:
    //{multiply(*this, A, alg_ideal_cast(order(A.which_base())));
    //  return *this;}

void assign(const module & A)
  {multiply(*this, A, alg_ideal_cast(order(A.which_base())));
    this->max_2exp = -1;
  }
  // original code:
  // {multiply(*this, A, alg_ideal_cast(order(A.which_base())));}

const bigint & first_generator() const { return gen1; }
const alg_number & second_generator() const { return gen2; }
void set_first_generator( const bigint & future_gen1 )
  { gen1 = future_gen1; }
void set_second_generator( const alg_number & future_gen2 ) {
    // Test: Do the current ideal and future_gen2 belong to
    // the same order?
       ...
    gen2 = future_gen2;
  }

// return the maximal precomputed 2-power
const int & get_max_2exp() const { return max_2exp; }

const bool powers_are_available() const {
    if ( max_2exp > 0 ) return (true); else return (false); }

// if 2-power ideal^{2^two_exp} available return first generator
// of that ideal
const bigint & first_generator_2power( const int & two_exp );

// if 2-power ideal^{2^two_exp} available return second generator
// of that ideal
  const alg_number & second_generator_2power( const int & two_exp );
```

```
// given the ideal in standard LiDIA representation compute two
// generators
void compute_generators();

// compute all 2-powers of ideal until 2^upper_bound
void compute_2powers( int upper_bound );

// multiply ideal in LiDIA representation with ideal in two element
// representation
friend void multiply_two_elt(alg_ideal &, const alg_ideal &,
    const alg_ideal &);

// determine a representative of the ideal class [A]^k using
// left-to-right exponentiation; use 2-element representation
// whenever possible
friend void power_reduce(alg_ideal &, const alg_ideal &, const bigint &);

// determine a representative of the ideal class [A]^k using
// fixed based windowing exponentiation; use 2-element representation
// whenenver posible
friend void power_reduce_fbw(alg_ideal &, const alg_ideal &, bigint &,
    const short & );

// transform a LiDIA representation of an ideal to a 2-element repres.
friend void zbasis_2_2elt( int,  bigint &, alg_number &,
    const alg_ideal &);

// output
friend ostream& operator<<(ostream &, const alg_ideal &);
void full_output();
```

5.2.3 Addition von Idealen und Division durch Ideale

Die von uns verwendete Addition und Division von Moduln und Idealen in LiDIA-Darstellung (und somit auch die Invertierung von Idealen) sind detailliert in [Nei02, S. 67ff.] beschrieben.

5.2.4 Test der Gleichheit zweier Gruppenelemente

Seien zwei Gruppenelemente $[\mathfrak{a}], [\mathfrak{b}]$ in der Klassengruppe eines algebraischen Zahlkörpers K vom Grad n gegeben. Die Entscheidung, ob diese beiden Elemente gleich sind, erweist sich als nichttriviales Problem, welches für betragsmäßig große Diskriminanten überhaupt nur in Klassengruppen algebraischer Zahlkörper mit sehr kleinem Regulator effizient gelöst werden kann. Allerdings haben beliebig gewählte algebraische Zahlkörper im Allgemeinen einen sehr großen Regulator ([Coh95]). In den NF-Kryptoverfahren aus Kapitel 3 muss der Beweiser bzw. Verifizierer bei vielen Identifikations- bzw. Signaturverfahren effizient entscheiden, ob zwei Gruppenelemente gleich sind oder nicht. In Kapitel 4 haben wir daher für die NF-Kryptographie nur Familien von Zahlkörpern

mit sehr kleinem Regulator vorgeschlagen. Für diese erweist sich die im Folgenden geschilderte Vorgehensweise zum Entscheid der Gleichheit zweier Gruppenelemente als effizient.

Quadratische Zahlkörper $(n = 2)$ Wenn K imaginär-quadratisch ist, so ist der Regulator $R(K) = 1$, es gibt genau ein reduziertes Ideal in jeder Klasse, und es existiert ein Polynomzeitalgorithmus zur Reduktion der Ideale. Gleichheit wird wie folgt entschieden: Reduziere zwei beliebige Vertreter $\mathfrak{a}$, $\mathfrak{b}$ der gegebenen Idealklassen. Es gilt $[\mathfrak{a}] = [\mathfrak{b}]$ genau dann, wenn die reduzierten Ideale gleich sind. Alternativ könnte man auch die für andere Zahlkörper gewählte Vorgehensweise (Hauptidealtest, siehe unten) anwenden: Es gilt $[\mathfrak{a}] = [\mathfrak{b}]$ genau dann, wenn $[\mathfrak{a}\mathfrak{b}^{-1}] = [\mathcal{O}]$, d.h. wenn $\mathfrak{a}\mathfrak{b}^{-1}$ ein Hauptideal $\alpha\mathcal{O}$ mit $\alpha \in K$ ist.

Wenn K ein reell-quadratischer Körper ist, dann können alle reduzierten Ideale, die äquivalent zu $\mathcal{O}$ sind, berechnet werden, sofern der Regulator $R(K)$ klein ist (siehe z.B. [Coh95]). Gleichheit wird dann entschieden, indem das Ideal $\mathfrak{a}\mathfrak{b}^{-1}$ reduziert wird und getestet wird, ob das Ergebnis mit einem der zu $\mathcal{O}$ äquivalenten reduzierten Ideale (d.h. mit einem reduzierten Hauptideal) übereinstimmt.

Zahlkörper vom Grad $n > 2$ Wir gehen wie für reell-quadratische Zahlkörper vor. Allerdings erweist es sich nun als viel schwieriger, alle reduzierten Hauptideale zu berechnen. Wir beschreiben im Folgenden Buchmanns Algorithmus [Buc87b]. Dieser stellt eine geometrische Verallgemeinerung des Kettenbruchalgorithmus von Lagrange dar. Der Algorithmus wurde ursprünglich entworfen, um Fundamentaleinheiten in Zahlkörpern zu bestimmen. Wir erläutern Pfahlers Modifikationen, dessen Optimierungen und Implementierung [Pfa03] zur Berechnung aller reduzierten Ideale einer Idealklasse. Die Implementierung und damit der Gleichheitsentscheid erweisen sich in der Praxis nur als effizient, wenn der Regulator $R(K)$ sehr klein ist ([Pfa03]). Zahlkörper mit nur einem reduzierten Hauptideal erlauben eine hinsichtlich des Gleichheitsentscheids besonders effiziente Implementierung der NF-Kryptoverfahren.

Wir diskutieren nun den Algorithmus zur Entscheidung der Gleicheit zweier Idealklassen. Dabei gehen wir wie folgt vor: Zunächst führen wir die notwendigen Begriffe ein. Anschließend erklären wir die grundlegende Vorgehensweise im Algorithmus; wir werden sehen, dass man im Wesentlichen einen sog. Minimazyklus in $[\mathcal{O}]$ zu berechnen hat. Anschließend erläutern wir, mit welchen Schwierigkeiten man bei der Implementierung zu kämpfen hat. Wir schildern Pfahlers Lösungen hierzu. Schließlich berichten wir über Pfahlers praktische Erfahrungen über Zeit- und Platzeffizienz. Wir orientieren uns im Folgenden eng an der Darstellung [Pfa03]. Wir betrachten einen algebraischen Zahlkörper K vom Grad $n > 2$ mit Signatur (r, s) und eine Ordnung $\mathcal{O}$ von K.

5.2.4.1 Minima, reduzierte Ideale und Einheiten

In diesem Unterabschnitt führen wir die notwendigen Begriffe ein.

Minima Sei $\mathfrak{a}$ ein Ideal in $\mathcal{O}$. Eine algebraische Zahl $\mu \in \mathfrak{a} - \{0\}$ heißt *Minimum von* $\mathfrak{a}$, wenn es keine algebraische Zahl $\alpha \in \mathfrak{a} - \{0\}$ gibt mit $|\alpha|_i < |\mu|_i$ für alle $1 \leqslant i \leqslant r + s$. Zwei Minima μ, ν von $\mathfrak{a}$ heißen *benachbart*, wenn es keine algebraische Zahl $\alpha \in \mathfrak{a} - \{0\}$ gibt mit $|\alpha|_i < \max\{|\mu|_i, |\nu|_i\}$ für alle $1 \leqslant i \leqslant r + s$. Ein maximales System nichtassoziierter Minima eines Ideals $\mathfrak{a}$ heißt *Minimazyklus* von $\mathfrak{a}$. Alle Minimazyklen von $\mathfrak{a}$ sind endlich; sie besitzen die

gleiche Kardinalität, die nach oben durch cR beschränkt ist, wobei R der Regulator von K ist und die positive Konstante c nur von der Signatur von K abhängt ([Buc87b, Buc87c]).

Reduzierte Ideale Ein Ideal $\mathfrak{a}$ in $\mathcal{O}$ heißt *reduziert*, wenn 1 ein Minimum von $\mathfrak{a}$ ist. Die Menge aller reduzierten Ideale in der Klasse von $\mathfrak{a}$ heißt *Zyklus der reduzierten Ideale* von $\mathfrak{a}$. Ihre Anzahl ist endlich und heißt *Periodenlänge* von $\mathfrak{a}$.

Zusammenhang zwischen Einheiten, Minima und reduzierten Idealen In einem reduzierten Ideal sind alle Einheiten Minima. Somit gibt es einen engen Zusammenhang zwischen der Berechnung von Fundamentaleinheiten und der Berechnung von Minima in Idealen. Thiel beschreibt und analysiert in [Thi95] die folgende Methode zur Idealreduktion: Finde ein Minimum im Ideal und dividiere das Ideal durch dieses Minimum. Genauer gilt sogar ([Buc87b, Buc87c]): Die reduzierten Ideale in der Klasse von $\mathfrak{a}$ sind genau die Ideale $\mu^{-1}\mathfrak{a}$, wobei μ einen Minimazyklus von $\mathfrak{a}$ durchläuft. Die Periodenlänge von $\mathfrak{a}$ stimmt mit der Länge eines beliebigen Minimazyklus von $\mathfrak{a}$ überein.

Die Gleichheit zweier Idealklassen $[\mathfrak{a}], [\mathfrak{b}]$ entscheiden wir, indem wir ein reduziertes Ideal in der Klasse von $\mathfrak{a}\mathfrak{b}^{-1}$ berechnen und prüfen, ob es mit einem der reduzierten Ideale der Klasse $[\mathcal{O}]$ übereinstimmt. Die reduzierten Ideale in $[\mathcal{O}]$ bestimmen wir durch Berechnung eines Minimazyklus $\{1, \mu_2, \ldots, \mu_k\}$ von $[\mathcal{O}]$: Die Menge aller reduzierten Ideale in $[\mathcal{O}]$ ist dann $\{\mathcal{O}, \mu_2^{-1}\mathcal{O}, \ldots, \mu_k^{-1}\mathcal{O}\}$.

5.2.4.2 Berechnung eines Minimazyklus nach Buchmann

Den Minimazyklus eines reduzierten Ideals $\mathfrak{a}$ kann man berechnen, indem man zu jedem schon bekannten Minimum von $\mathfrak{a}$ (am Anfang kennt man das Minimum 1) die endliche Menge aller benachbarten Minima berechnet und überprüft, ob man echt neue Minima erhalten hat, also solche Minima, die nicht assoziiert zu bereits bekannten Minima sind. Diesen Schritt wiederholt man so lange, bis keine echt neuen Minima mehr gefunden werden. Diese Nachbarnsuche funktioniert, weil der ungerichtete Graph, der durch Verbinden der benachbarten Minima (nichtassoziierte Minima sind Knoten des Graphen) mit Kanten entsteht, zusammenhängend ist ([Pfa03]).

Für benachbarte Minima μ, ν in $\mathfrak{a}$ ist $\nu\mu^{-1}$ Nachbar von 1 im reduzierten Ideal $\mu^{-1}\mathfrak{a}$ und umgekehrt ([Pfa03]). Somit kann man die Suche nach allen Nachbarn eines Minimums μ im Ideal $\mathfrak{a}$ auf die Suche nach allen Nachbarn von 1 in reduzierten Idealen $\mu'^{-1}\mathfrak{a}$ zurück führen:

Zunächst berechnet man alle Nachbarn von 1 in $\mathfrak{a}$. Zu jedem Nachbarn μ, der den bislang ermittelten Minimazyklus vergrößert, bestimmt man dann im reduzierten Ideal $\mu^{-1}\mathfrak{a}$ alle Nachbarn von 1. Dieser Schritt wird für alle auf dem Minimazyklus neu gefundenen Minima so lange wiederholt, bis keine echt neuen Minima (also solche, die nichtassoziiert zu bereits auf dem Minimazyklus gefundenen Minima sind) gefunden werden.

Wir geben nun den exakten Algorithmus zur Berechnung eines Minimazyklus nach Buchmann [Buc87b] in der Darstellung nach [Pfa03] an:

Algorithmus 5.4 Berechnung eines Minimazyklus

Eingabe: Ein reduziertes Ideal $\mathfrak{a}$.

Ausgabe: Ein Minimazyklus von $\mathfrak{a}$.

1: $k := 1$, $p := 1$, $\mathfrak{b}_1 := \mathfrak{a}$, $\mu_1 := 1$
2: **while** $k \leqslant p$ **do**
3: Berechne die Menge N aller Nachbarn von 1 in $\mathfrak{b}_k$
4: **for** $\eta \in N$ **do**
5: $\mathfrak{b} := \eta^{-1}\mathfrak{b}_k$, $\mu := \eta\mu_k$
6: **if** $\mathfrak{b} \neq \mathfrak{b}_j$ für alle $j \in \{1, \ldots, p\}$ **then** $p := p + 1$, $\mathfrak{b}_p := \mathfrak{b}$, $\mu_p := \mu$
7: **end for**
8: $k := k + 1$
9: **end while**
10: **return** $\{\mu_1, \ldots, \mu_k\}$

Wir haben noch zu erklären, wie man alle Nachbarn von 1 in einem reduzierten Ideal berechnen kann. Wir erklären Buchmanns Methode [Buc87b], die auf dem Begriff der minimalen Teilmengen eines Ideals basiert.

Minimale Teilmengen, Expansionen und Kompressionen Sei S eine endliche nichtleere Teilmenge des Ideals $\mathfrak{a}$. Wir definieren:

1. Für $i \in \{1, \ldots, r+s\}$ sei

$$|S|_i = \max\{|\alpha|_i \mid \alpha \in S\}$$
$$S(i) = \{\alpha \in S \mid |\alpha|_i = |S|_i \text{ und } |\alpha|_j < |S|_j \text{ für } 1 \leqslant j \leqslant r+s, j \neq i\} \ .$$

2. S heißt *minimale Teilmenge* (von $\mathfrak{a}$), wenn $S = \{\alpha \in \mathfrak{a} \mid 0 < |\alpha|_i \leqslant |S|_i \text{ für } 1 \leqslant i \leqslant r+s\}$ und wenn es kein $\alpha \in \mathfrak{a} - \{0\}$ gibt mit $|\alpha|_i < |S|_i$ für $1 \leqslant i \leqslant r+s\}$.

3. Zu jeder minimalen Teilmenge S und jedem Index $i \in \{1, \ldots, r+s\}$ gibt es genau eine minimale Teilmenge S', so dass $|S'|_j = |S|_j$ für $1 \leqslant j \leqslant r+s$, $j \neq i$ und $S'(i)$ nichtleer. S' wird *i-te Expansion* von S genannt, in Zeichen $S' = e_i(S)$.

4. Für $i \in \{1, \ldots, r+s\}$ heißt $k_i(S) = \{\alpha \in S \mid |\alpha|_i < |S|_i\}$ die *i-te Kompression* von S. Sofern $k_i(S)$ nichtleer ist, ist $k_i(S)$ auch minimale Teilmenge.

Alle Elemente einer minimalen Teilmenge S eines Ideals $\mathfrak{a}$ sind Minima in $\mathfrak{a}$. Außerdem ist mit $\mu \in S$ auch $\mu\xi \in S$ für alle Einheitswurzeln ξ.

Unterprogramm zur Nachbarnsuche der 1 In einem reduzierten Ideal $\mathfrak{a}$ gibt es zu jedem Nachbarn μ von 1 eine minimale Teilmenge S von $\mathfrak{a}$, die 1 und μ enthält. Um alle Nachbarn von 1 zu bestimmen, genügt es daher, alle minimalen Teilmengen von $\mathfrak{a}$ zu berechnen, die 1 enthalten.

Buchmann beweist in [Buc87b] folgenden Zusammenhang: Je zwei minimale Teilmengen S und S' von $\mathfrak{a}$ sind durch eine endliche Folge $S = S_1, S_2, \ldots, S_k = S'$ von minimalen Teilmengen verbunden, wobei S_j entweder eine Expansion oder eine Kompression von S_{j-1} ist $(2 \leqslant j \leqslant k)$.

Insgesamt erhält man also alle benachbarten Minima der 1 in einem reduzierten Ideal durch eine geeignete Folge von Expansionen und Kompressionen einer minimalen Teilmenge des Ideals.

Wir geben nun den exakten (Unter-)Algorithmus zur Berechnung aller Nachbarn der 1 in einem reduzierten Ideal gemäß [Buc87b] in der Darstellung nach [Pfa03] an:

Algorithmus 5.5 Berechnung aller Nachbarn von 1

Eingabe: Ein reduziertes Ideal $\mathfrak{b}$.
Ausgabe: Die Menge aller Nachbarn von 1 in $\mathfrak{b}$.

1: $S_1 := \{\alpha \in \mathfrak{b} \mid 0 < |\alpha|_i \leqslant 1 \text{ für } 1 \leqslant i \leqslant r+s\}$
2: $p := 1$, $k := 1$
3: **while** $k \leqslant p$ **do**
4: **for** $1 \leqslant i \leqslant r+s$ **do**
5: $S := e_i(S_k)$
6: **if** $S \neq S_j$ für alle $1 \leqslant j \leqslant p$ **then** $p := p+1$, $S_p := S$
7: **if** $|S_k|_i > 1$ **then**
8: $S := k_i(S)$
9: **if** $S \neq S_j$ für alle $1 \leqslant j \leqslant p$ **then** $p := p+1$, $S_p := S$
10: **end if**
11: **end for**
12: $k := k+1$
13: **end while**
14: **return** $\bigcup_{j=1}^{p} S_j$

Bei Eingabe von $\mathfrak{b} = \mathcal{O}$ ist die in Schritt (1) zu berechnende Menge S_1 gerade die Menge der Einheitswurzeln in K ([Pfa03]). Diese können einfach im Computer berechnet werden (siehe [Coh95]).

Die Korrektheit und Terminierung der von uns angegebenen Algorithmen werden in [Buc87b] bewiesen.

5.2.4.3 Pfahlers Implementierung und Optimierungen

Pfahler [Pfa03] implementierte die beschriebenen Algorithmen 5.4 und 5.5 und löste die dabei auftretenden Schwierigkeiten. Zudem optimierte er Algorithmus 5.5 bei der wiederholten Berechnung aller benachbarten Minima der 1 in reduzierten Idealen.

Implementierungsprobleme und Pfahlers Lösungen Die Hauptschwierigkeiten bei der Implementierung der angegeben Algorithmen lauten wie folgt: Man muss algebraische Zahlen mit kleinen archimedischen Bewertungen finden. Ferner hat man zu entscheiden, welche Größenrelation ($<, =, >$) zwischen zwei Bewertungen algebraischer Zahlen gilt. Hierzu gibt es keine direkte Lösung. Pfahler löst diese Schwierigkeiten durch Anwendung der Minkowski-Abbildung auf die algebraischen Zahlen des Zahlkörpers, d.h. durch Übertragen des Problems in den $\mathbb{R}^n$. [1] Reelle Zahlen können im Computer aber nur approximiert werden. Pfahler analysiert insbesondere, wie

[1] Diese Idee wurde bereits in [Buc88] formuliert und in [Thi95] komplexitätstheoretisch untersucht. Pfahler [Pfa03] verfeinert die Güte der notwendigen Approximationen.

genau diese Approximationen sein müssen, damit die Algorithmen garantiert korrekte Resultate liefern. Pfahler gibt in Computern direkt implementierbare Algorithmen an, insbesondere zur Kompression und Expansion minimaler Teilmengen von reduzierten Idealen.

Pfahlers Optimierungen Pfahler optimiert den beschriebenen Buchmann-Algorithmus bei der wiederholten Berechnung aller benachbarten Minima der 1 in reduzierten Idealen:

1. Pfahler vereinfacht drastisch die wiederholte Berechnung der minimalen Teilmengen reduzierter Ideale, mit denen im Teilalgorithmus 5.5 jeweils begonnen wird. Pfahler gelingt es hier, bereits gesammelte Berechnungsergebnisse bei späteren Anwendungen von Algorithmus 5.5 auszunutzen.

2. Pfahler beweist Rechenregeln für etliche Kombinationen aus Kompressionen und Expansionen minimaler Teilmengen eines Ideals und reduziert durch deren Anwendung den Rechenaufwand beträchtlich.

5.2.4.4 Experimentelle Resultate

Wir geben nun einige experimentelle Resultate für Pfahlers Implementierung [Pfa03] wieder. Wir schlüsseln Laufzeiten und den Speicherplatzbedarf für den Algorithmus auf und geben die berechnete Periodenlänge von $\mathcal{O}$ an. Die Untersuchungen wurden auf einem Intel Pentium III mit 600 MHz Taktfrequenz durchgeführt. Als Compiler kam GNU egcs-1.1.2 zum Einsatz.

Wir betrachten Stender-Zahlkörper $K = \mathbb{Q}(\sqrt[n]{a})$ mit $a = D^n + d$, $d \in \{\pm 1\}$, $D \in \mathbb{Z}_{>0}$, $n \in \{3, 4, 6\}$.

Tabelle 5.4 enthält die experimentellen Resultate. Die ersten vier Spalten spezifizieren den untersuchten Stender-Körper: Zahlkörpergrad (erste Spalte), Bitlänge der Diskriminante (zweite Spalte), Parameter D und d (dritte und vierte Spalte). Die fünfte Spalte enthält die Periodenlänge von $\mathcal{O}$. Die sechste Spalte gibt die Laufzeit zur Berechnung aller reduzierten Hauptideale an. (Hierbei schließen wir nicht die Zeit zur Berechnung einer Ganzheitsbasis der Maximalordnung ein, denn diese wird zur Durchführung der kryptographischen Protokolle ohnehin benötigt.) Die letzte Spalte gibt den im Arbeitsspeicher benötigten Speicherplatz an. Das Programm selbst nimmt knapp 2MB ein. Die Tabelle zeigt folgendes:

1. Stender-Körper sind hinsichtlich eines effizienten Gleichheitsentscheids für Idealklassen sehr geeignet, da die Periodenlänge von $\mathcal{O}$ jeweils sehr klein ist. In Pfahlers Experimenten war die Periodenlänge für den Grad 3 stets kleiner als 3, für den Grad 4 stets kleiner als 6 und für den Grad 6 stets kleiner als 40.

2. Pfahlers Implementierung der Berechnung aller reduzierten Hauptideale einer Ordnung verhilft zu einem Gleichheitsentscheid für Idealklassen, der für kryptographische Anwendungen hinreichend effizient ist, sofern mit geeigneten Zahlkörpern gearbeitet wird: Die Programmlaufzeiten sind hinreichend kurz, (insbesondere für die Grade 3,4), und auch für betragsmäßig große Diskriminanten wird wenig Speicherplatz benötigt.

n	$\log \lvert\Delta\rvert$	**D**	**d**	**P**	**t**	**M**
3	364	1056636907677369006	+1	1	<1s	<2MB
3	364	1056636907677369006	-1	2	<1s	<2MB
3	412	270499048365406465526	+1	1	<1s	<2MB
3	412	270499048365406465526	-1	2	<1s	<2MB
3	494	4835703278458516698824748	+1	1	<1s	<2MB
3	494	4835703278458516698825342	-1	2	<1s	<2MB
3	566	19807040628566084398385988822	+1	1	<1s	<2MB
3	566	19807040628566084398385988540	-1	2	<1s	<2MB
3	692	41538374868278621028243970633762338	+1	1	<1s	<2MB
3	692	41538374868278621028243970633771020	-1	2	<1s	<2MB
3	806	217780 ... 714829400616616559748756331655482 52	+1	1	<1s	<2MB
3	806	217780 ... 714829400616616559748756331655391 68	-1	2	<1s	<2MB
3	1160	125542034707733615276715 ... 788464153328322047108889280690425 66	+1	1	1.9s	<3MB
3	1160	125542034707733615276715 ... 788464153328322047108889280690370 94	-1	2	2.5s	<3MB
4	336	169103741	+1	3	<1s	<3MB
4	336	169103756	-1	4	<1s	<3MB
4	379	2026954653	+1	3	<1s	<3MB
4	379	2026954658	-1	4	<1s	<3MB
4	455	163443381348	+1	5	1s	<3MB
4	455	163443381348	-1	4	1s	<3MB
4	528	11082382755514	+1	5	1s	<3MB
4	528	11082382755514	-1	4	1s	<3MB
4	634	5055119662503616	+1	5	1s	<3MB
4	634	5055119662503620	-1	4	1.2s	<3MB
4	733	2054272581141342540	+1	5	2s	<3MB
4	738	2054272581141342534	-1	4	1.4s	<3MB
4	1062	27571979853348664 1868378984	+1	5	3s	<3MB
4	1062	27571979853348664 1868378978	-1	4	2.3s	<3MB
6	336	1880	+1	33	36s	<3MB
6	379	5090	+1	33	39s	<3MB
6	455	29517	+1	39	49s	<3MB
6	528	159465	+1	39	54s	<3MB
6	634	1846389	+1	39	61s	<3MB
6	738	20412387	+1	39	70s	3.1 MB
6	1062	36392991094	+1	33	106s	3.5 MB

Tabelle 5.4: Gleichheitstest für Idealklassen, experimentelle Resultate

5.2.5 Zufällige Auswahl eines Gruppenelementes

In den meisten der NF-Kryptoverfahren muss man Gruppenelemente (Idealklassen) zufällig und gleichverteilt auswählen. Bis jetzt war unklar, wie dies von statten gehen soll. Algorithmus 5.6 beschreibt unsere Lösung dieses Auswahlproblems.

Algorithmus 5.6 Zufällige und gleichverteilte Wahl einer Idealklasse

Eingabe: Ein algebraischer Zahlkörper K.

Ausgabe: Eine zufällig und gleichverteilt ausgewählte Idealklasse $[\mathfrak{a}]$.

1: Vorberechnung: Berechne als Erzeugendensystem von $Cl(K)$ die Menge $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$ der Primideale (exakt: Klassen der Primideale) mit Norm $\leqslant 12 \ln^2 |\Delta|$.

2: Wähle einen zufälligen Exponentenvektor $(e_1, \ldots, e_k)$ mit $1 \leqslant e_i \leqslant |\Delta|$ $(1 \leqslant i \leqslant k)$.

3: Berechne $\mathfrak{a} := \text{LLL} - \text{reduce}\left(\prod_{i=1}^{k} \mathfrak{p}_i^{e_i}\right)$.

4: **return** $[\mathfrak{a}]$

Wir zeigen, dass man mit Algorithmus 5.6 ein Gruppenelement in $Cl(K)$ zufällig und „fast“ gleichverteilt auswählen kann. Dazu beweisen wir zunächst den folgenden Satz.

Satz 5.2.1
Seien $\mathcal{G}$ ein Erzeugendensystem von $Cl(K)$, $[\mathfrak{a}] \in Cl(K)$. Dann ist die Anzahl der Exponentenvektoren $r = (r([\mathfrak{p}]))_{[\mathfrak{p}]\in\mathcal{G}} \in \{1, 2, \ldots, |\Delta|\}^{|\mathcal{G}|}$ mit $\prod_{[\mathfrak{p}]\in\mathcal{G}} [\mathfrak{p}]^{r([\mathfrak{p}])} = [\mathfrak{a}]$ gleich

$$\frac{|\mathcal{G}|^{|\Delta|}}{h(K)} \cdot exp(\varepsilon_{\mathfrak{a}}) \ ,$$

wobei $\varepsilon_{\mathfrak{a}} \in \mathbb{R}$, $|\varepsilon_{\mathfrak{a}}| < \frac{h(K)}{|\Delta| - h(K)} < 1$. Für hinreichend großes $|\Delta|$ gilt $|\varepsilon_{\mathfrak{a}}| \leqslant \ln 2$, d.h. $0.5 \leqslant exp(\varepsilon_{\mathfrak{a}}) \leqslant 2$..

In Satz 5.2.1 ergäbe sich eine Gleichverteilung, wenn die Anzahl der Exponentenvektoren r nicht von der speziellen Wahl einer Idealklasse $[\mathfrak{a}] \in Cl(K)$ abhängen würde. Die Abweichung unserer Verteilung von der Gleichverteilung wird durch die Konstante $exp(\varepsilon_{\mathfrak{a}})$ ausgedrückt. Diese hängt von der speziellen Wahl von $[\mathfrak{a}]$ ab. Je näher $exp(\varepsilon_{\mathfrak{a}})$ an 1 liegt, umso mehr nähert sich die Verteilung bei unserer Auswahl eines Gruppenelementes der Gleichverteilung.

Beweis: Der Beweis des Satzes 5.2.1 erfolgt vollkommen analog zum Beweis von [LP92, Theorem 5.2], welches die Formulierung des Satzes 5.2.1 für den Spezialfall imaginär-quadratischer Zahlkörper ist. □

Korollar 5.2.2
Algorithmus 5.6 wählt in Polynomzeit ein Gruppenelement pseudozufällig aus der Klassengruppe aus. Die Abweichung der Verteilung von der Gleichverteilung ist wie folgt charakterisiert: Sei $[\mathfrak{a}] \in Cl(K)$ beliebig. Dann ist die Anzahl der Exponentenvektoren $r = (r([\mathfrak{p}]))_{[\mathfrak{p}]\in\mathcal{G}} \in \{1, 2, \ldots, |\Delta|\}^{|\mathcal{G}|}$ mit $\prod_{[\mathfrak{p}]\in\mathcal{G}} [\mathfrak{p}]^{r([\mathfrak{p}])} = [\mathfrak{a}]$ gleich

$$\frac{|\mathcal{G}|^{|\Delta|}}{h(K)} \cdot exp(\varepsilon_{\mathfrak{a}}) \ ,$$

wobei $\varepsilon_{\mathfrak{a}} \in \mathbb{R}$, $|\varepsilon_{\mathfrak{a}}| < \frac{h(K)}{|\Delta| - h(K)} < 1$. Für hinreichend großes $|\Delta|$ gilt $|\varepsilon_{\mathfrak{a}}| \leqslant \ln 2$, d.h. $0.5 \leqslant exp(\varepsilon_{\mathfrak{a}}) \leqslant 2$..

Beweis: Bach [Bac90] zeigt unter Annahme der verallgemeinerten Riemannschen Vermutung, dass die Menge der Primideale mit Norm $\leqslant 12\ln^2|\Delta|$ ein Erzeugendensystem der Klassengruppe $Cl(K)$ ist. Satz 5.2.1 liefert nun unmittelbar die Behauptung. □

Parameterwahl in der Praxis. In der Praxis verfolgen wir bei der Auswahl eines Gruppenelementes die konkurrierenden Ziele Qualität des Pseudozufalls und Effizienz:

1. Je kleiner wir die Menge der Primideale machen, umso effizienter können wir ein Potenzprodukt bilden, also ein Gruppenelement auswählen; jedoch wird die Auswahl weniger zufällig, denn die Wahrscheinlichkeit, nur einen Teil (eine echte Untergruppe) der Klassengruppe zu erzeugen, wird immer größer.

2. Je kleiner wir die Exponenten e_i wählen, umso effizienter bilden wir das Potenzprodukt zur Auswahl eines Gruppenelementes; jedoch wird die Auswahl weniger zufällig, denn die Wahrscheinlichkeit, auf diese Weise nur einen Teil aller Elemente der Klassengruppe auswählen zu können, wird immer größer.

Zu 1. In Abhängigkeit von der Signatur des Körpers sind bessere Schranken als $\leqslant 12\ln^2|\Delta|$ für die Norm bekannt, wenn es um die Auswahl der Primideale für ein Erzeugendensystem der Klassengruppe geht. Beispielsweise genügt im Falle von imaginär-quadratrischen Zahlkörpern die Wahl aller Primideale mit Norm $\leqslant 6\ln^2|\Delta|$. In der Praxis können wir sogar viel kleinere Normschranken benutzen, um ein Erzeugendensystem zu bilden: Die Experimente von Neis [Nei02] zeigen, dass die Wahl von Normschranken in der Größenordnung $\ln^{1.5}|\Delta|$ für Klassengruppen von Zahlkörpern der Grade $3, 4, 6$ genügt.

Zu 2. Bei der Wahl der Exponenten genügt es für die theoretische Analyse, Exponenten in der Größenordnung der Klassenzahl (also $\approx |\Delta|^{0.45}$ für kubische Stender-Körper, siehe Abschnitt 4.4.2.2) zu wählen. In der kryptographischen Praxis genügt es nach heutigem Kenntnisstand, deutlich kleinere zufällige Exponenten zu wählen.

5.3 Implementierung und Laufzeiten der NF-Kryptoverfahren

In diesem Abschnitt beschreiben wir zunächst einige Besonderheiten unserer Implementierung von Kryptoverfahren über Klassengruppen algebraischer Zahlkörper. Dann untersuchen wir die Komplexität der NF-Kryptoverfahren und deren Laufzeiten.

5.3.1 Implementierung

Wir implementierten folgende der in Kapitel 3 vorgestellten kryptographischen Protokolle in Klassengruppen algebraischer Zahlkörper:

1. Signaturverfahren
   (a) RDSA
   (b) DSA-No-Subgroup
   (c) R-Feige-Fiat-Shamir (R-FFS)

2. Verschlüsselungsverfahren

   (a) Diffie-Hellman Integrated Encryption Scheme (DHIES)

3. Identifikationsverfahren

   (a) R-Fiat-Shamir (R-FS)

Wir verwendeten dabei die Programmiersprachen C und C++ sowie das Computeralgebrasystem LiDIA [LiD01], welches wir zur Effizienzsteigerung erweiterten.

Als Hashalgorithmus benutzten wir `RIPEMD-160` (siehe [MOV97, S. 349ff.]), welches für kryptographische Zwecke als geeignet angesehen wird. Zur Durchführung einer effizienten Exponentiation in den Klassengruppen (notwendig für RDSA, DSA-No-Subgroup, Guillou-Quisquater, DHIES, nicht aber für R-Feige-Fiat-Shamir und R-Fiat-Shamir) kombinierten wir unser neues Verfahren zur Idealmultiplikation mit der bekannten Exponentiationsmethode *Fixed Base Windowing* zur Basis 16 (siehe [MOV97]). Hierzu berechneten wir alle benötigten 16-Potenzen des Basiselementes (Idealklasse) γ vor und speicherten jeweils den LLL-reduzierten Vertreter der entsprechenden Idealklasse in Zwei-Element-Darstellung. Somit nutzten wir zum Aufmultiplizieren unsere neue effiziente Methode zur Idealmultiplikation (siehe Abschnitt 5.2.2.2). Zur LLL-Reduktion der Ideale wendeten wir die Schnorr-Euchner-Variante [SE91] des LLL-Algorithmus an. Die geschilderte schnelle Exponentiationsmethode für Idealklassen haben wir als Elementfunktion `power_reduce_fbw` der C++-Klasse `alg_ideal` in LiDIA implementiert (siehe Unterabschnitt 5.2.2). Für die meisten Idealmultiplikationen ($> 70\%$) steht uns dabei ein Ideal in Zwei-Element-Darstellung zur Verfügung, so dass wir unsere schnelle Idealmultiplikationsmethode nutzen können. Für die restlichen Idealmultiplikationen wenden wir den Standard-LiDIA-Algorithmus an (siehe Unterabschnitt 5.2.2.2).

Wir analysieren die Komplexität der Fixed Base Windowing-Exponentiationsmethode zur Berechnung von γ^k für ein gegebenes Element $\gamma \in G$ und einen Exponenten $k \approx 2^m$:
Sei $b \in \mathbb{Z}_{\geqslant 2}$ die verwendete Basiszahl. Die durchschnittliche Anzahl an Gruppenoperationen (average case) beträgt $\frac{b-1}{b}\lceil \log_b k \rceil + b - 3$. Hierbei kann in $\frac{b-1}{b}(\lceil \log_b k \rceil - 1)$ vielen Gruppenoperationen unsere neue Idealmultiplikation angewendet werden, für die restlichen $\frac{b-1}{b} + b - 3$ Gruppenoperationen muss die Standard-Multiplikationsmethode angewendet werden. Die maximale Anzahl an Gruppenoperationen (worst case) ist $\lceil \log_b k \rceil + b - 3$. Hiervon kann in $\lceil \log_b k \rceil - 1$ vielen Gruppenoperationen unsere neue Idealmultiplikation angewendet werden, für die restlichen $b - 2$ Gruppenoperationen muss die Standard-Multiplikationsmethode angewendet werden. Es müssen jeweils $\lceil \log_b k \rceil$ viele Gruppenelemente gespeichert werden.

In den meisten der von uns implementierten kryptographischen Protokolle müssen wir Potenzen von Gruppenelementen mit unterschiedlich großen Exponenten berechnen. Beispielsweise müssen wir im RDSA-Signaturverfahren eine Exponentiation eines Gruppenelementes mit einem Exponenten $k \approx 2^{160} \cdot |G|$ und eine Exponentiation mit einem Exponenten $\lambda \approx 2^{160}$ durchführen. Bei Implementierung des RDSA-Verfahrens in Stender-Körpern vom Grad 3 (bzw. vom Grad 4) entspricht eine Diskriminante $\Delta \approx -2^{404}$ (bzw. $\Delta \approx -2^{382}$) sicherheitstechnisch einem RSA-1024 Bit-Modulus (siehe Abschnitt 4.5). Für Stender-Körper vom Grad 3 mit 404-Bit-Diskriminante (bzw. Stender-Körper vom Grad 4 mit 382-Bit-Diskriminante) erwarten wir die Klassenzahl in der Größenordnung $h \approx |\Delta|^{0.45} \approx 2^{182}$ (bzw. $h \approx |\Delta|^{0.41} \approx 2^{157}$). Zur Erstellung einer RDSA-Signatur muss man also in Stender-Körpern vom Grad 3 mit 404-Bit-Diskriminante (bzw. in

Tabelle 5.5: Komplexitätsanalyse Fixed Base Windowing mit Exponent $k \approx 2^{342}$

Basis-zahl	Durchschnittl. #Gruppenop. #total	$\frac{\#\text{LiDIA}\cdot\text{2Elt.}}{\#\text{total}}$	Maximale #Gruppenop. #total	$\frac{\#\text{LiDIA}\cdot\text{2Elt.}}{\#\text{total}}$	Speicherpl. (#Elemente)
2	170	100%	341	100%	342
4	129.25	98.6%	172	98.8%	171
8	104.75	94.4%	119	95.0%	114
16	93.63	85.1%	99	85.9%	86
32	95.84	68.7%	98	69.4%	69
64	117.11	47.1%	118	47.5%	57

Stender-Körpern vom Grad 4 mit 382-Bit-Diskriminante) neben der 160-Bit-Exponentiation eine 342-Bit- (bzw. 317-Bit-)Exponentiation durchführen. Die beiden Tabellen 5.5, 5.6 geben in Abhängigkeit von der gewählten Basiszahl b Aufschluss über die Komplexität der Fixed Base Windowing-Methode bei einer 342-Bit und einer 160-Bit-Exponentiation (mit zufällig gewähltem Exponenten in der entsprechenden Größenordnung). In der ersten Spalte findet sich jeweils die Basiszahl b, die zweite Spalte enthält die durchschnittlich notwendige Anzahl auszuführender Gruppenoperationen, die dritte Spalte enthält den Anteil an Gruppenoperationen, bei denen unsere neue Idealmultiplikationsmethode anwendbar ist. Die vierte und fünfte Spalte enthalten die entsprechenden Werte für die maximal notwendige Anzahl an Gruppenoperationen. Die sechste Spalte gibt die Anzahl der Gruppenelemente an, die zur Ausführung des Algorithmus gespeichert werden müssen. Die Tabellen zeigen, dass für die beiden Exponentiationen bei Wahl der Basiszahl $b = 16$ in der Fixed Base Windowing-Exponentiationsmethode durchschnittlich am wenigsten Gruppenoperationen durchzuführen sind. Wir empfehlen daher die Wahl von $b = 16$. Tabelle 5.7 gibt bei Verwendung von $b = 16$ Aufschluss über die Komplexität einer Fixed Base Windowing-Exponentiation mit dem Exponenten k, dessen Bitlänge durch die erste Spalte ausgedrückt wird. Der übrige Aufbau der Tabelle stimmt mit den Tabellen 5.5, 5.6 überein. Bei einer 342-Bit-Exponentiation können die Gruppenoperationen durchschnittlich in 85.1% der Fälle mit unserer neuen Methode zur Idealmultiplikation kombiniert werden. Bei einer 160-Bit-Operation beträgt der Anteil der Gruppenoperationen, bei denen die neue Idealmultiplikationsmethode angewendet werden kann, 72.4%.

In der Praxis erhalten Benutzer des RDSA-Signaturverfahrens[2] den öffentlichen Schlüssel des Senders einer signierten Nachricht sowie die vorberechneten 16-Potenzen des Basiselementes γ. Jeder Benutzer rechnet in der gleichen Klassengruppe $G = Cl(K)$, mit dem gleichen Basiselement γ und mit den gleichen 16-Potenzen von γ zur Erstellung eigener Signaturen.

5.3.2 Komplexität und Laufzeiten

In diesem Abschnitt analysieren wir zunächst die Komplexität einiger Kryptoverfahren über Klassengruppen algebraischer Zahlkörper. Anschließend untersuchen wir experimentell deren Laufzei-

[2]sowie anderer Signaturverfahren, bei denen der Signierer eine Exponentiation mit möglicher Teilvorberechnung durchzuführen hat

Tabelle 5.6: Komplexitätsanalyse Fixed Base Windowing mit Exponent $k \approx 2^{160}$

Basis-zahl	Durchschnittl. #Gruppenop. #total	$\frac{\#\text{LiDIA·2Elt.}}{\#\text{total}}$	Maximale #Gruppenop. #total	$\frac{\#\text{LiDIA·2Elt.}}{\#\text{total}}$	Speicherpl. (#Elemente)
2	79	100%	159	100%	160
4	61	97.1%	81	97.5%	80
8	52.25	88.8%	59	89.8%	54
16	50.5	72.4%	53	73.6%	40
32	60	50.0%	61	50.8%	32
64	87.58	29.2%	88	29.5%	27

Tabelle 5.7: Komplexitätsanalyse Fixed Base Windowing mit Basiszahl 16 und variablem Exponenten k

$\lceil \log k \rceil$	Durchschnittl. #Gruppenop. #total	$\frac{\#\text{LiDIA·2Elt.}}{\#\text{total}}$	Maximale #Gruppenop. #total	$\frac{\#\text{LiDIA·2Elt.}}{\#\text{total}}$	Speicherpl. (#Elemente)
160	50.5	72.4%	53	73.6%	40
$160+0.41 \cdot 337 \approx 299$	83.3	83.3%	88	84.1%	75
$160+0.41 \cdot 382 \approx 317$	88.0	84.2%	93	84.9%	80
$160+0.45 \cdot 358 \approx 322$	88.9	84.3%	94	85.1%	81
$160+0.45 \cdot 404 \approx 342$	93.6	85.1%	99	85.9%	86
$160+0.41 \cdot 525 \approx 376$	101.1	86.2%	107	86.9%	94
400	106.8	86.9%	115	87.6%	100
$160+0.45 \cdot 550 \approx 408$	108.6	87.2%	117	87.8%	102
$160+0.41 \cdot 641 \approx 423$	112.4	87.6%	119	88.2%	106
$160+0.41 \cdot 740 \approx 464$	121.8	88.6%	129	89.1%	116
$160+0.45 \cdot 668 \approx 461$	121.8	88.6%	129	89.1%	116
$160+0.45 \cdot 769 \approx 507$	132.1	89.4%	140	90.0%	127

ten.

Komplexität der Berechnungen

RDSA Bei der Durchführung der Operationen dominieren die Gruppenoperationen (Exponentiationen) zeitlich. Andere Operationen (z.B. Anwenden einer Hashfunktion) fallen zeitlich nicht ins Gewicht. Die Berechnung einer RDSA-Signatur erfordert zwei Exponentiationen in der zugrunde liegenden Gruppe (Klassengruppe), wobei eine 160-Bit-Exponentiation und eine Exponentiation mit einem Exponenten $k \approx 2^{160} \cdot |G|$ durchgeführt wird. Zur Verifikation einer Signatur müssen drei Exponentiationen durchgeführt werden, wobei hier effiziente Algorithmen zur simultanen Exponentiation angewendet werden können (siehe [MOV97]). Zudem müssen zur Verifikation zwei Gruppenoperationen und ein Tabellen-Look-Up für einen Gleichheitsentscheid durchgeführt werden.

Bei Verwendung einer Diskriminante $\Delta \approx -2^{404}$ eines kubischen Stender-Körpers (bzw. einer Diskriminante $\Delta \approx -2^{382}$ eines Stender-Körpers vom Grad 4) [3] ist $h = |G| \approx |\Delta|^{0.45}$, also $|G| \approx 2^{182}$ (bzw. $h = |G| \approx |\Delta|^{0.41} \approx 2^{157}$), so dass wir hier zur Signaturberechnung eine 160-Bit- und eine 342-Bit-Exponentiation (bzw. eine 160-Bit- und eine 317-Bit-Exponentiation) durchzuführen haben. Den Tabellen 5.5, 5.6 und 5.7 entnehmen wir, dass zur Signaturerstellung bei Verwendung der Fixed Base Windowing-Exponentiationsmethode mit Basiszahl 16 durchschnittlich 144.1 (bzw. 138.5) viele Gruppenoperationen durchzuführen sind, wobei in 80.6% (bzw. 79.9%) der Fälle unsere neue effiziente Idealmultiplikationsmethode angewendet werden kann. Im ungünstigsten Fall (worst case) sind hier 152 (bzw. 146) viele Gruppenoperationen durchzuführen, wobei in 81.6% (bzw. 80.8%) der Fälle unsere neue effiziente Idealmultiplikationsmethode angewendet werden kann. Insgesamt benötigen wir Speicherplatz für 86 (bzw. 80) Gruppenelemente. (Bei beiden Exponentiationen wird die gleiche Idealklasse als Basis verwendet!)

DSA-No-Subgroup Auch beim Signieren oder Verifizieren einer Signatur gemäß DSA-No-Subgroup dominieren die Gruppenoperationen (Exponentiationen) zeitlich. Andere Operationen (z.B. Anwenden einer Hashfunktion) fallen zeitlich nicht ins Gewicht. Die Berechnung einer DSA-No-Subgroup-Signatur erfordert eine 400-Bit-Exponentiation in der zugrunde liegenden Gruppe (Klassengruppe). Diese kann komplett vorberechnet werden, falls ein zufällig ausgewählter Exponent k vorab zur Verfügung steht. Die Erstellung einer Signatur kann in diesem Fall in einer Millisekunde durchgeführt werden. Kann der zufällige Exponent erst zur Laufzeit gewählt werden, können geeignete Potenzen des Basiselementes γ vorberechnet und die Fixed Base Windowing-Methode mit unserer schnellen Idealmultiplikationsmethode eingesetzt werden. Die Verifikation einer DSA-No-Subgroup-Signatur erfordert zwei Exponentiationen (mit 400-Bit- bzw. 160-Bit-Exponent), eine weitere Gruppenoperation und einen Gleichheitsentscheid (Tabellen-Look-Up). Hierbei können effiziente Algorithmen zur simultanen Exponentiation angewendet werden (siehe [MOV97]).

Der Tabelle 5.7 entnehmen wir: Bei Verwendung der von uns empfohlenen Parameter und Wahl des zufälligen Exponenten $k \in \{1, \ldots, 2^{400}\}$ zur Laufzeit müssen wir zur Signaturerstellung bei Verwendung der Fixed Base Windowing-Exponentiationsmethode mit Basiszahl 16 durchschnittlich 106.8 viele Gruppenoperationen durchführen, wobei in 86.9% der Fälle

[3] dies entspricht der Sicherheit von RSA-1024 Bit, siehe Abschnitt 4.5

unsere neue effiziente Idealmultiplikationsmethode angewendet werden kann. Im ungünstigsten Fall (worst case) sind hier 113 viele Gruppenoperationen durchzuführen, wobei in 87.6% der Fälle unsere neue effiziente Idealmultiplikationsmethode angewendet werden kann. Insgesamt benötigen wir Speicherplatz für 100 Gruppenelemente.

Guillou-Quisquater-Signaturverfahren (GQ) Die Berechnung einer GQ-Signatur erfordert die Auswahl eines zufälligen Gruppenelementes (Vorberechnung ist möglich), zwei 160-Bit-Exponentiationen und eine weitere Gruppenoperation. Eine der beiden Exponentiationen kann komplett vorberechnet werden, falls vorab eine zufällig zu wählende Idealklasse zur Verfügung steht; andernfalls ist keinerlei Vorberechnung möglich. Bei der anderen Exponentiation können geeignete Potenzen des privaten Schlüsels α vorberechnet werden, so dass die Fixed Base Windowing-Methode mit unserer schnellen Idealmultiplikation eingesetzt werden kann. Die Verifikation einer GQ-Signatur erfordert zwei 160-Bit-Exponentiationen, eine weitere Gruppenoperation und einen Gleichheitsentscheid. Hierbei können effiziente Algorithmen zur simultanen Exponentiation angewendet werden (siehe [MOV97]).

Der Tabelle 5.7 entnehmen wir: Bei Verwendung der von uns empfohlenen Parameter (öffentlicher 160-Bit-Exponent und 160-Bit-Hashwerte) und Wahl der zufälligen Idealklasse κ zur Laufzeit müssen wir zur Signaturerstellung bei Verwendung der Fixed Base Windowing-Exponentiationsmethode mit Basiszahl 16 durchschnittlich 262 viele Gruppenoperationen durchführen[4]. Hinzu kommt der Aufwand zur Auswahl einer zufälligen Idealklasse. Damit ist klar, dass die Erstellung von GQ-Signaturen deutlich ineffizienter als die Signaturerstellung gemäß der vorher diskutierten Verfahren ist. Daher vertiefen wir die GQ-Analyse an dieser Stelle nicht weiter.

R-Feige-Fiat-Shamir-Signaturverfahren (R-FFS) Auch hier dominieren die Gruppenoperationen die zeitliche Komplexität (siehe die Erläuterungen für RDSA weiter oben). Die Berechnung einer R-FFS-Signatur erfordert die Auswahl eines zufälligen Gruppenelementes und die Berechnung einer k-ten Potenz in der zugrunde liegenden Gruppe (Klassengruppe); beides kann komplett vorberechnet werden. Zudem sind durchschnittlich $\frac{m}{2}$ (im ungünstigsten Fall m) viele Gruppenoperationen durchzuführen. Hierbei kann unsere neue Idealmultiplikationsmethode angewendet werden. Zur Verifikation einer Signatur muss der gleiche Aufwand betrieben werden, wobei kein zufälliges Gruppenelement auszuwählen ist und die k-te Potenz nicht vorab berechnet werden kann.

Bei Verwendung der von uns empfohlenen Parameter $m = 80$ und $k = 2$ (bzw. $k = 3$) sind folglich zur Signaturerstellung und Verifikation jeweils durchschnittlich 41 (bzw. 42) und im ungünstigsten Fall 81 (bzw. 82) Gruppenoperationen durchzuführen. Hinzu kommt der Aufwand zur Auswahl einer zufälligen Idealklasse. Der Signierer kann für alle bis auf eine Gruppenoperation (zwei Gruppenoperationen für $k = 3$) die schnelle Idealmultiplikationsmethode einsetzen; der Verifizierer kann ohne Ausnahme die schnelle Idealmultiplikationsmethode verwenden.

DHIES-Verschlüsselungsverfahren (DHIES) Zur Verschlüsselung sind zwei Exponentiationen mit Exponenten in der Größenordnung der Gruppenordnung durchzuführen. Dies kann komplett als Vorberechnung erfolgen, wobei unsere neue Idealmultiplikationsmethode angewendet werden kann. Das Ableiten einer Bitsequenz aus zwei Gruppenelementen (die Routine *deriveKey*, welche wir mittels des Hashalgorithmus `RIPEMD-160` implementierten) sowie die

[4](Berechnung der 16-Potenzen (160), zwei Exponentiationen $(2 \cdot 50.5)$ und eine Gruppenoperation)

Durchführung eines symmetrischen Verschlüsselungsverfahrens (3DES) und das (optionale) Berechnen eines Message Authentication Codes sind vergleichbar einfache Bitoperationen, die zeitlich nicht ins Gewicht fallen. Gleiches gilt für die zum Verschlüsseln notwendige Wahl einer Zufallszahl. Analog wird die Entschlüsselung im DHIES-Verfahren durch eine Exponentiation mit einem Exponenten in der Größenordnung der Gruppenordnung dominiert. Vorberechnung ist hier nicht möglich, und unsere neue Idealmultiplikationsmethode kann nicht eingesetzt werden.

Bei Verwendung einer Diskriminante $\Delta \approx -2^{404}$ eines kubischen Stender-Körpers (bzw. einer Diskriminante $\Delta \approx -2^{382}$ eines Stender-Körpers vom Grad 4) [5] ist $h = |G| \approx |\Delta|^{0.45}$, also $|G| \approx 2^{182}$ (bzw. $h = |G| \approx |\Delta|^{0.41} \approx 2^{157}$), so dass wir hier zur Verschlüsselung zwei 182-Bit-Exponentiationen (bzw. zwei 157-Bit-Exponentiationen) durchzuführen haben; zur Entschlüsselung ist nur eine 182-Bit- (bzw. 157-Bit-)Exponentiation notwendig. Bei Verwendung der Fixed Base Windowing-Exponentiationsmethode mit Basiszahl 16 sind zur Verschlüsselung durchschnittlich 112.3 (bzw. 101.0) viele Gruppenoperationen durchzuführen, zur Entschlüsselung sind es 56.1 (bzw. 50.5) viele Gruppenoperationen. Dabei kann in 75.2% (bzw. 72.4%) der Fälle unsere neue effiziente Idealmultiplikationsmethode angewendet werden.

R-Fiat-Shamir-Identifikationsverfahren (R-FS) Der Beweiser **A** hat t Gruppenelemente zufällig zu wählen und t viele k-te Potenzen zu berechnen. Hierbei ist eine Vorberechnung komplett möglich. Zudem hat **A** im Durchschnitt $\frac{mt}{2}$ (im ungünstigsten Fall mt) viele Gruppenoperationen auszuführen. Hier ist keine Vorberechnung möglich, aber die schnelle Idealmultiplikationsmethode kann eingesetzt werden. Der Verifizierer **B** hat ebenfalls t viele k-te Potenzen zu berechnen und im Durchschnitt $\frac{mt}{2}$ (im ungünstigsten Fall mt) viele Gruppenoperationen auszuführen. Zudem hat der Verifizierer t viele Gleichheitsentscheide durchzuführen.

Bei Verwendung der von uns empfohlenen Parameter $t = 1$ und $m = 20$ sind folglich für $k = 2$ (bzw. $k = 3$) für Beweiser **A** und Verifizierer **B** jeweils durchschnittlich 11 (bzw. 12) und im ungünstigsten Fall 21 (bzw. 22) Gruppenoperationen durchzuführen. Sowohl Beweiser als auch Verifizierer können für alle bis auf eine (im Fall $k = 3$ zwei) Gruppenoperationen die schnelle Idealmultiplikationsmethode einsetzen. Der Beweiser hat zusätzlich eine zufällige Idealklasse auszuwählen.

Laufzeiten von RDSA, DSA-No-Subgroup, R-FFS, R-FS, DHIES

Wir untersuchen nun experimentell die Laufzeiten unserer Kryptoverfahren.

Ziel unserer Arbeit ist insbesondere der praktische Nachweis, dass man Kryptoverfahren in algebraischen Zahlkörpern auch höheren Grades implementieren kann. Daher hielten wir unsere Implementierung unabhängig vom Zahlkörpergrad. Für spezielle Zahlkörpergrade optimierte Implementierungen scheinen möglich und sind Gegenstand künftiger Forschung. Diese werden signifikante Effizienzverbesserungen bringen. In unserer Implementierung wird der größte Zeitanteil auf die Reduktion von Idealen verwendet (ca. 90%, siehe unten). Eine Beschleunigung der Idealreduktion wird daher unmittelbar die Durchführung der kryptographischen Protokolle beschleunigen. Wir präsentieren einige Ideen:

[5] dies entspricht der Sicherheit von RSA-1024 Bit, siehe Abschnitt 4.5

Tabelle 5.8: Systemumgebung für alle Laufzeitmessungen

CPU:	Celeron-Prozessor mit Taktfrequenz 433MHz
Betriebssystem:	Linux (Kernel 2.2.12)
Compiler:	GNU Project C und C++-Compiler (gcc/ g++ Version 2.95)
Externe Bibliotheken:	LiDIA Version 2.0 (algebraische Zahlkörper)
	OpenSSL Version 0.9.6g (RSA, `http://www.openssl.org`)

Tabelle 5.9: Laufzeiten für eine Gruppenoperation

n=3 $\lceil \log \lvert\Delta\rvert \rceil$	n=3 LiDIA $*$ LiDIA	n=3 LiDIA $*$ 2-Elt	n=4 $\lceil \log \lvert\Delta\rvert \rceil$	n=4 LiDIA $*$ LiDIA	n=4 LiDIA $*$ 2-Elt
358	14.9 ms	13.5 ms	337	40.2 ms	37.0 ms
404	17.8 ms	16.1 ms	382	47.5 ms	42.9 ms
550	28.8 ms	26.3 ms	525	75.4 ms	68.7 ms
668	39.5 ms	36.2 ms	641	104.8 ms	95.1 ms
769	52.9 ms	47.5 ms	740	138.1 ms	124.2 ms

Die Idealreduktion wird man in kubischen Zahlkörpern vermutlich beschleunigen können, indem man die Voronoi-Reduktion [Vor96] (siehe auch [HW94]) anwendet. Van Sprangs FTPR-Algorithmus [van94] der paarweisen Gauß-Reduktion könnte für kleine Körpergrade > 2 ebenfalls effizienter als LLL-Reduktion sein. Für große Zahlkörpergrade könnte Schnorrs Block-LLL-Reduktionsalgorithmus [KS01a, KS01b] effizienter als die von uns verwendete Schnorr-Euchner-Variante des LLL-Algorithmus sein.

Tabelle 5.8 beschreibt die Systemumgebung, in der alle in diesem Kapitel genannten Laufzeiten gemessen wurden.

Tabelle 5.9 zeigt die Laufzeiten unserer Implementierung für eine Gruppenoperation in Klassengruppen von Stender-Körpern. Die erste und vierte Spalte beschreiben die Parametergrößen der Klassengruppe: den Zahlkörpergrad n und die binäre Länge der Diskriminante Δ. Die Spalten 2 und 5 geben die Laufzeit für eine Gruppenoperation unter Verwendung der LiDIA-Implementierung an; die Spalten 3 und 6 beinhalten die entsprechende Laufzeit unter Verwendung unserer neuen Idealmultiplikation. Die Laufzeiten sind in Millisekunden angegeben und über 1000 Iterationen gemittelt.

Die folgenden Tabellen 5.10, 5.11, 5.12, 5.13 und 5.14 zeigen die Laufzeiten unserer Implementierungen der Signaturverfahren RDSA, DSA-No-Subgroup, R-FFS, des Verschlüsselungsverfahrens DHIES und des Identifikationsverfahrens R-FS in Stender-Körpern der Grade 3 und 4. (Das GQ-Signaturverfahren ist ineffizienter als RDSA, DSA-No-Subgroup und R-FFS; wir werden daher dessen Laufzeit nicht untersuchen.) In den meisten realen Anwendungen muss die Erstellung von Signaturen besonders effizient erfolgen; wir geben daher lediglich die Laufzeiten für die Signaturerstellung an.

Die ersten beiden Spalten von Tabelle 5.10 geben die Bitlänge eines RSA-Modulus und die Zeit

zur Erstellung einer RSA-Signatur unter Verwendung der Bibliothek OpenSSL (siehe Tabelle 5.8) an. Die dritte Spalte (bzw. siebte Spalte) gibt die Größe der sicherheitstechnisch korrespondierenden Diskriminante eines Stender-Körpers vom Grad 3 (bzw. vom Grad 4) an. Die Spalten 4,5 und 6 (bzw. 8, 9 und 10) beziehen sich auf das Erstellen einer RDSA-Signatur im Stender-Körper dritten (bzw. vierten) Grades: Spalte 4 (bzw. 8) gibt die durchschnittliche Anzahl durchzuführender Gruppenoperationen an. Spalte 5 (bzw. 9) beschreibt, zu welchem Anteil hierbei unsere neue schnelle Idealmultiplikation angewendet werden kann. Spalte 6 (bzw. 10) zeigt, in welcher Zeit (in Millisekunden) wir eine RDSA-Signatur erstellen. Dabei verwenden wir die in Abschnitt 3.3.4.1 empfohlenen Parameter, um ein beweisbares Sicherheitsniveau[6] zu erlangen. Die Laufzeiten sind über 1000 Iterationen gemittelt.

Die anschließend folgenden Tabellen 5.11, 5.12, 5.13 und 5.14 sind analog aufgebaut; sie drücken die Laufzeiten anderer von uns implementierter kryptographischer Verfahren über Stender-Körpern der Grade 3 und 4 aus. Wir verwendeten jeweils die von uns in Kapitel 3 empfohlenen Parameter. In Tabelle 5.13 verwenden wir bei der RSA-Verschlüsselung den öffentlichen Exponenten $e = 2^{16} + 1 = 65537$, so dass eine Potenzierung zum Exponenten e extrem effizient erfolgt.

Einige Verfahren erfordern die zufällige Auswahl eines Gruppenelementes (R-FFS, R-FS) oder einer Zufallszahl (RDSA, DSA-No-Subgroup, DHIES). In manchen realen Einsatzszenarien ist es möglich, solche zufälligen Daten vorab zu erzeugen, in anderen nicht. Bei den angegebenen Laufzeiten gehen wir für das Signaturverfahren R-FFS und das Identifikationsverfahren R-FS (optimistisch) davon aus, dass die Zufallsdaten bereits vorab erstellt wurden. Hingegen gehen wir bei RDSA, DSA-No-Subgroup und DHIES (pessimistisch) vom Gegenteil aus. Wir bemerken erneut, dass eine DSA-No-Subgroup-Signatur in einer Millisekunde durchgeführt werden kann, falls ein zufällig ausgewählter Exponent k vorab zur Verfügung steht, da dann auch die Exponentiation γ^k vorab erfolgen kann. Ähnlich effizient würde die Verschlüsselung von Texten mit DHIES erfolgen. Die Erstellung einer RDSA-Signatur würde sich im Wesentlichen auf die Durchführung einer 160-Bit-Exponentiation, also auf durchschnittlich 50.5 Gruppenoperationen reduzieren. Hingegen würden die Verfahren R-FFS für den Signierer und R-FS für den Beweiser unattraktiver werden, wenn sie zur Laufzeit jeweils ein zufälliges Gruppenelement erzeugen müssten. In der Praxis wären dann alternative Verfahren wie das DSA-No-Subgroup-Signaturverfahren und das R-Schnorr-Identifikationsprotokoll (siehe Kapitel 3) zu bevorzugen.

Wir stellen fest: Kryptographie über algebraischen Zahlkörpern ist heute bereits ein wichtiges Backup zu den heute in der Praxis verwendeten Verfahren RSA, DSA und auf elliptischen Kurven basierenden Verfahren. Denn wir entnehmen den Tabellen 5.10, 5.11, 5.12, 5.13 und 5.14:

1. Auf Zahlkörperkryptographie basierende Identifikationsverfahren sind heute bereits hinsichtlich ihrer zeitlichen Effizienz praxistauglich. Beispielsweise benötigen Beweiser wie Verifizierer im R-FS-Identifikationsprotokoll auf veralteter PC-Hardware bei Verwendung von Parametern, die ein hohes Sicherheitsniveau versprechen, deutlich weniger als eine halbe Sekunde.

2. Heute steht bereits eine Implementierung eines auf Zahlkörpern basierenden Signaturverfahrens (R-FFS) zur Verfügung, welche auf leistungsschwacher PC-Hardware die Erstellung einer nach heutigen Gesichtspunkten sicheren Signatur in weniger als einer Sekunde erlaubt.

[6] (basierend auf gewissen Annahmen)

Tabelle 5.10: Laufzeiten für RDSA-Signaturverfahren in Stender-Körpern

RSA	RSA sign	n=3 $\lceil \log \lvert\Delta\rvert \rceil$	#Gr.op.	Ant. eff. Idealm.	RDSA sign	n=4 $\lceil \log \lvert\Delta\rvert \rceil$	#Gr.op.	Ant. eff. Idealm.	RDSA sign
768	10 ms	358	139.4	80.0%	1921 ms	337	133.8	79.2%	5041 ms
1024	15 ms	404	144.1	80.6%	2386 ms	382	138.5	79.9%	6038 ms
2048	95 ms	550	159.1	82.5%	4308 ms	525	151.6	81.6%	10606 ms
3072	308 ms	668	172.3	83.9%	6395 ms	641	162.9	82.9%	15768 ms
4096	640 ms	769	182.6	84.7%	9004 ms	740	172.3	83.9%	18339 ms

Tabelle 5.11: Laufzeiten für DSA-No-Subgroup-Signaturverfahren in Stender-Körpern

RSA	RSA sign	n=3 $\lceil \log \lvert\Delta\rvert \rceil$	DSA-No-Subgroup sign	n=4 $\lceil \log \lvert\Delta\rvert \rceil$	DSA-No-Subgroup sign
768	10 ms	358	1460 ms	337	3993 ms
1024	15 ms	404	1746 ms	382	4648 ms
2048	95 ms	550	2845 ms	525	7435 ms
3072	308 ms	668	3919 ms	641	8976 ms
4096	640 ms	769	5158 ms	740	13471 ms

3. Würde ein traditionelles Verschlüsselungsverfahren wie RSA gebrochen, stünde heute bereits mit DHIES über algebraischen Zahlkörpern eine sichere Alternative zur Verfügung, die Textblöcke auf alter PC-Hardware in weniger als 2 Sekunden verschlüsselt.

4. Mit zunehmender Steigerung der Sicherheit (durch Vergrößerung der Schlüssellängen) wird der heute noch existente Effizienznachteil der NF-basierten Verfahren im Vergleich zu RSA immer kleiner. Während die Erstellung einer R-FFS-Signatur (bzw. einer RDSA-Signatur) in kubischen Stender-Körpern für kleinere Schlüssellängen (RSA-768) noch um den Faktor 67 (bzw. Faktor 192) langsamer ist, verringert sich der Laufzeitunterschied für etwas größere Schlüssellängen (RSA-4096) bereits auf den Faktor 3 (bzw. den Faktor 14).

5. Durch Verwendung heute marktüblicher moderner PCs mit mehr als 2 GHz Taktfrequenz reduzieren sich die Laufzeiten unserer Kryptoverfahren auf weniger als 20% der von uns in den Tabellen angegebenen Zeiten.

Tabelle 5.12: Laufzeiten für R-FFS-Signaturverfahren in Stender-Körpern

RSA	RSA sign	n=3 $\lceil \log \lvert\Delta\rvert \rceil$	R-FFS sign	n=4 $\lceil \log \lvert\Delta\rvert \rceil$	R-FFS sign
768	10 ms	358	561 ms	337	1524 ms
1024	15 ms	404	668 ms	382	1769 ms
2048	95 ms	550	1089 ms	525	2828 ms
3072	308 ms	668	1494 ms	641	3907 ms
4096	640 ms	769	1957 ms	740	5108 ms

Tabelle 5.13: Laufzeiten für DHIES-Verschlüsselungsverfahren in Stender-Körpern

RSA	RSA encrypt	n=3 $\lceil \log \lvert\Delta\rvert \rceil$	DHIES encrypt	n=4 $\lceil \log \lvert\Delta\rvert \rceil$	DHIES encrypt
768	<1 ms	358	1423 ms	337	3482 ms
1024	<1 ms	404	1896 ms	382	4390 ms
2048	3 ms	550	3918 ms	525	8932 ms
3072	6 ms	668	6341 ms	641	14522 ms
4096	12 ms	769	9492 ms	740	21327 ms

Tabelle 5.14: Laufzeiten für R-FS-Identifikationsverfahren in Stender-Körpern

RSA	RSA sign	n=3 $\lceil \log \lvert\Delta\rvert \rceil$	R-FS prove/ verify	n=4 $\lceil \log \lvert\Delta\rvert \rceil$	R-FS prove/ verify
768	10 ms	358	148 ms	337	408 ms
1024	15 ms	404	179 ms	382	475 ms
2048	95 ms	550	294 ms	525	759 ms
3072	308 ms	668	399 ms	641	1049 ms
4096	640 ms	769	528 ms	740	1380 ms

Das Durchführen einer Gruppenoperation in Klassengruppen algebraischer Zahlkörper besteht aus einer Multiplikation von Idealen, gefolgt von der Reduktion des Idealproduktes. Unsere experimentellen Untersuchungen (Profiling) zeigen, dass zum Durchführen einer Gruppenoperation auf die Multiplikation der Ideale höchstens etwa 10%, für die Reduktion des Idealproduktes mindestens etwa 90% der Gesamtzeit verwendet wird. Diese Daten beziehen sich auf Stender-Körper der Grade 3 und 4 unabhängig von der Größe der Diskriminante. Zur weiteren signifikanten Steigerung der zeitlichen Effizienz von NF-Kryptosystemen muss man demnach

- die Anzahl der durchzuführenden Gruppenoperationen minimieren (z.B. durch Konstruktion effizienterer Kryptosysteme)
- oder die Idealreduktion beschleunigen.

Literaturverzeichnis

[18694] FIPS 186 (Federal Information Procesing Standards Publication 186). Digital signature standard. U.S. Department of Commerce/N.I.S.T., National Technical Information Service, 1994.

[95995] ISO/IEC 9594-8. Information technology - open systems interconnection - the directory: Authentication framework. International Organization for Standardization (equivalent to ITU-T Recommendation X.509, 1993), 1995.

[ABR01] M. Abdalla, M. Bellare, and P. Rogaway. An encryption scheme based on the diffie-hellman problem. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 01*, pages 143–158, 2001.

[ACGS88] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM Journal on Computing*, 17:194–209, 1988.

[AT91] K. Alagappan and J. Tardo. SPX guide: Prototype public-key authentication service. Digital Equipment Corp., 1991.

[Bac90] E. Bach. Explicit bounds for primality testing and related problems. *Math. Comp.*, 55:355–380, 1990.

[BBB+02] C. Balut, K. Belabas, D. Bernardi, H. Cohen, and M. Olivier. *A User's Guide to PARI/GP*. Université Bordeaux I, France, 2002. Available at `ftp://megrez/math.u-bordeaux.fr/pub/pari/`.

[BBHM99] J. Buchmann, I. Biehl, S. Hamdy, and A. Meyer. Cryptographic protocols based on the intractability of extracting roots and computing discrete logarithms. Technical report No. TI-16/99, Darmstadt University of Technology, 1999.

[BBHM00] J. Buchmann, I. Biehl, S. Hamdy, and A. Meyer. A signature scheme based on the intractability of computing roots. Technical report No. TI-1/00, Darmstadt University of Technology, 2000.

[BBHM02] J. Buchmann, I. Biehl, S. Hamdy, and A. Meyer. A signature scheme based on the intractability of extracting roots. *Design, Codes and Cryptography*, 25(3):223–236, 2002.

[BBT94] I. Biehl, J. Buchmann, and C. Thiel. Cryptographic protocols based on discrete logarithms in real-quadratic orders. In *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 56–60. Springer, 1994.

[BCDP91] J. Boyar, D. Chaum, I. Damgard, and T. Pedersen. Convertible undeniable signatures. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology — CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 1991.

[BD90] J. Buchmann and S. Düllmann. On the computation of discrete logarithms in class groups. In *Advances in Cryptology — CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 134–139. Springer, 1990.

[BD95] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–288. Springer, 1995.

[BDW90] J. Buchmann, S. Düllmann, and H.C. Williams. On the complexity and efficiency of a new key exchange system. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 597–616. Springer, 1990.

[BH01] J. Buchmann and S. Hamdy. A survey on IQ cryptography. In K. Alster, J. Urbanowicz, and H.C. Williams, editors, *Public Key Cryptography and Computational Number Theory, 2000*, pages 1–15. deGruyter, 2001.

[BJT97] J. Buchmann, M.J. Jacobson, Jr., and E. Teske. On some computational problems in finite abelian groups. *Mathematics of Computation*, 66:1663–1687, 1997.

[BLP93] J.P. Buhler, H.W. Lenstra, Jr., and C. Pomerance. Factoring integers with the number field sieve. volume 1554 of *Lecture Notes in Mathematics*, pages 50–94. Springer, 1993.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BNPS01] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA inversion problems and the security of Chaum's blind signature scheme. IACR eprint archive report No. 2001/002, http://eprint.iacr.org/2001/002/. Preliminary version The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme in Financial Cryptography '01, Lecture Notes in Computer Science Vol. 2339, P. Syverson ed., Springer, 2001.

[BP97] J. Buchmann and S. Paulus. A one way function based on module arithmetic in number fields. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, number 1294 in Lecture Notes in Computer Science, pages 385–394. Springer, 1997.

[BP02] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In M. Yung, editor, *Advances in Cryptology — CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS66] Z.I. Borevic and I.R. Safarevic. *Number theory*. Academic Press, New York, 1966.

[BSW90] J. Buchmann, R. Scheidler, and H.C. Williams. Implementation of a key exchange protocol using real quadratic fields. In I.B. Damgard, editor, *Advances in Cryptology — EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 98–109. Springer, 1990.

[BSW94] J. Buchmann, R. Scheidler, and H.C. Williams. A key-exchange protocol using real quadratic fields. *Journal of Cryptology*, 7, 1994.

[BTW95] J. Buchmann, C. Thiel, and H.C. Williams. Short representation of quadratic integers. In *Computational Algebra and Number Theory, Mathematics and its Applications*, volume 325, pages 159–185. Kluwer, Dordrecht, 1995.

[Buc87a] J. Buchmann. The computation of the fundamental unit of totally complex quartic orders. *Mathematics of Computation*, 48(177):39–54, January 1987.

[Buc87b] J. Buchmann. On the computation of units and class numbers by a generalization of Lagrange's algorithm. *Journal of Number Theory*, 26:8–30, 1987.

[Buc87c] J. Buchmann. On the period length of the generalized Lagrange algorithm. *Journal of Number Theory*, 26:31–37, 1987.

[Buc88] J. Buchmann. Zur Komplexität der Berechnung von Einheiten und Klassenzahlen algebraischer Zahlkörper. Habilitationsschrift, Universität Düsseldorf, Germany, 1988.

[Buc89] J. Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. In C. Goldstein, editor, *Séminaire de théorie des nombres*, volume 91 of *Progress in Mathematics*, pages 28–41. Birkhäuser, Paris, 1989.

[Buc91] J. Buchmann. Number theoretic algorithms and cryptology. In *FCT '91*, volume 529 of *Lecture Notes in Computer Science*, pages 16–21. Springer, 1991.

[BW88] J. Buchmann and H.C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1:107–118, 1988.

[BW89] J. Buchmann and H.C. Williams. A key-exchange system based on real quadratic fields. In *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 335–343. Springer, 1989.

[BW90] J. Buchmann and H.C. Williams. Quadratic fields and cryptography. In *Number Theory and Cryptography, London Math. Soc. Lecture Note Series*, volume 154, pages 9–26. Springer, 1990.

[Can97] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1997.

[CDEH+96] J. Cowie, B. Dodson, R.M. Elkenbracht-Huizing, A.K. Lenstra, P.L. Montgomery, and J. Zayer. A world wide number field sieve factoring record: On to 512 bits. In *Proc. of ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 382–394. Springer, 1996.

[CEvdG88] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In D. Chaum and W.L. Price, editors, *Advances in Cryptology — EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. Springer, 1988.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *13th Annual ACM Symposium on the Theory of Computing*, pages 209–218. ACM Press, 1998.

[Cha83] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology — CRYPTO '82*, Lecture Notes in Computer Science, pages 199–203. Plenum Press, 1983.

[Cha85] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28:1030–1044, 1985.

[Cha95] D. Chaum. Designated confirmer signatures. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer, 1995.

[CL83] H. Cohen and H.W. Lenstra, Jr. Heuristics on class groups of number fields. In *Number Theory, Lecture notes in Math.*, volume 1068, pages 33–62. Springer, New York, 1983.

[CM87] H. Cohen and J. Martinet. Class groups of number fields: numerical heuristics. *Math. Comp.*, (48):123–137, 1987.

[CM90] H. Cohen and J. Martinet. Etude heuristique des groupes de classes des corps de nombres. *J. Reine Angew. Math.*, (404):39–76, 1990.

[CMR98] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hashing. In *30th Annual ACM Symposium on the Theory of Computing*, pages 131–140. ACM Press, 1998.

[Coh95] H. Cohen. *A course in computational algebraic number theory*. Springer, Heidelberg, 1995.

[COS86] D. Coppersmith, A.M. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1:1–15, 1986.

[Cox89] D.A. Cox. *Primes of the form* $x^2 + ny^2$. Wiley, New York, 1989.

[CP93] D. Chaum and T. Pedersen. Wallet databases with observers. In E.F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1993.

[CPS95] J.L. Camenisch, J.-M. Piveteau, and M.A. Stadler. Blind signatures based on the discrete logarithm problem. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '84*, volume 950 of *Lecture Notes in Computer Science*, pages 428–432. Springer, 1995.

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, number 1294 in Lecture Notes in Computer Science. Springer, 1997.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[Cv90] D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer, 1990.

[CvH91] D. Chaum and E. van Heyst. Group signatures. In D.W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

[DGB88] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 1988.

[DM96] T.F. Denny and V. Müller. On the reduction of composed relations from the number field sieve. In *Proc. of Algorithmic Number Theory Symposium II (ANTS II)*, volume 1122 of *Lecture Notes in Computer Science*. Springer, 1996.

[DSW96] T. Denny, O. Schirokauer, and D. Weber. Discrete logarithms: the effectiveness of the index calculus method. In *Proc. of Algorithmic Number Theory Symposium II (ANTS II)*, volume 1122 of *Lecture Notes in Computer Science*. Springer, 1996.

[Dül91] S. Düllmann. *Ein Algorithmus zur Bestimmung der Klassengruppe positiv definiter binärer quadratischer Formen*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1991.

[DvOW92] W. Diffie, P.C. van Oorschot, and M.J. Wiener. Authentication and authenticated key exchanges. *Design, Codes and Cryptography*, 2:107–125, 1992.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1985.

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[FP03] P.-A. Fouque and G. Poupard. On the security of RDSA. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, Lecture Notes in Computer Science. Springer, 2003. To be published.

[FR94] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.

[GB01] S. Goldwasser and M. Bellare. Lecture notes on cryptography. `http://www-cse.ucsd.edu/users/mihir/papers/gb.html`, 2001.

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 1997.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *25th IEEE Symposium on the Foundations of Computer Science*. IEEE Press, 1984.

[GHR99] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer, 1999.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18:186–208, 1989.

[Gol99] O. Goldreich. Modern cryptography, probabilistic proofs and pseudorandomness. volume 17 of *Algorithms and Combinatorics*, New York, 1999. Springer.

[GQ88] L.C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. Günther, editor, *Advances in Cryptology — EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer, 1988.

[Ham02] S. Hamdy. *Über die Sicherheit und Effizienz kryptographischer Verfahren mit Klassengruppen imaginär-quadratischer Zahlkörper*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2002.

[Hel85] B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125–139, 1985.

[HJP98] D. Hühnlein, M.J. Jacobson, Jr., and S. Paulus. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In *Advances in Cryptology — EUROCRYPT '98*, page to appear, 1998.

[HM89] J. L. Hafner and K. S. McCurley. A rigorous subexponential algorithm for computation of class groups. *J. Amer. Math. Soc.*, 2:839–850, 1989.

[HM00] S. Hamdy and B. Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In *Proc. of ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 2000.

[HMNP99] T. Hahn, A. Meyer, S. Neis, and T. Pfahler. Implementing cryptographic protocols based on algebraic number fields. Technical report No. TI-24/99, Darmstadt University of Technology, 1999.

[HMP94] P. Horster, M. Michels, and H. Petersen. Meta-ElGamal signature schemes. In *Proc. 2nd ACM Conference on Computer and Communications Security*, pages 96–107. ACM Press, 1994.

[HMT98] D. Hühnlein, A. Meyer, and T. Takagi. Rabin and RSA analogues based on non-maximal imaginary quaratic orders. In *International Conference on Information Security and Cryptology - ICISC 98, Korea*, pages 221–240, 1998.

[How86] J.A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Lin. Mult. Alg.*, 19:67–77, 1986.

[HS97] S. Hunter and J. Sorenson. Approximating the number of intergers free of large prime factors. *Math. Comp.*, 66:1729–1741, 1997.

[HW79] G.H. Hardy and E.M. Wright. *An introduction to the theory of numbers*. Oxford University Press, Oxford, 1979.

[HW94] B.K. Schmid H.C. Williams, G.W. Dueck. A rapid method of evaluating the regulator and class number of a pure cubic field. *Math. Comp.*, 63:377–387, 1994.

[Kan87] R. Kannan. Minkowski's konvex body theorem and integer programming. *Mathematics of Operations Research*, 12, no. 5:415–440, 1987.

[Kob89] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.

[Kob90] N. Koblitz. A family of Jacobians suitable for discrete log cryptosystems. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 94–99. Springer, 1990.

[KS01a] H. Koy and C.P. Schnorr. Segment LLL-reduction of lattice bases. In J.H. Silverman, editor, *Cryptography and Lattices*, volume 2146 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2001.

[KS01b] H. Koy and C.P. Schnorr. Segment LLL-reduction with floating point orthogonalization. In J.H. Silverman, editor, *Cryptography and Lattices*, volume 2146 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2001.

[LiD01] LiDIA Group. *LiDIA - A library for computational number theory*. Technische Hochschule Darmstadt, Germany, 1994-2001. Available at `http://www.informatik.tu-darmstadt.de/TI/LiDIA`.

[LM78] D.H. Lehmer and J.M. Masley. Table of the cyclotomic class numbers $h^*(p)$ and their factors for $200 < p < 521$. *Math. Comp.*, 32:577–582, 1978.

[Lou94] S. Louboutin. The exponent 2-class-group problem for non-galois- over $\mathbb{Q}$ quartic fields that are quadratic extensions of imaginary quadratic fields. *J. Number Theory*, 49:133–141, 1994.

[Lou95] S. Louboutin. Class-number problems for cubic number fields. *Nagoya Math. J.*, 138:199–208, 1995.

[LP92] H.W. Lenstra Jr. and C. Pomerance. A rigorous time bound for factoring integers. *J. Amer. Math. Soc.*, 5:483–516, 1992.

[LV99] A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes in commercial applications. *CCE Quarterley Journal*, 3:3–10, 1999. Available at `http://www.cryptosavvy.com`.

[LV01] A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 4:255–293, 2001.

[McC89] K. McCurley. Cryptographic key distribution and computation in class groups. In R.A. Mollin, editor, *Number Theory and Applications*, pages 459–479. Kluwer Academic Publishers, 1989.

[Mey97] A. Meyer. Ein neues Identifikations- und Signaturverfahren über imaginär-quadratischen Zahlkörpern. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 1997. Available at `ftp://ftp.informatik.tu-darmstadt.de/pub/TI/reports/amy.diplom.ps.gz`.

[Mic93] S. Micali. Fair public-key cryptosystems. In E.F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 113–138. Springer, 1993.

[MNP01] A. Meyer, S. Neis, and T. Pfahler. Implementing cryptographic protocols based on algebraic number fields. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy, Proc. of ACISP 2001, the 6th Australasian Conference on Information Security and Privacy*, volume 2119 of *Lecture Notes in Computer Science*, pages 84–103. Springer, 2001.

[MOV91] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 80–89, 1991.

[MOV97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[MTI86] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *Transactions of the IECE (Japan)*, 69:99–106, 1986.

[Nei94] S. Neis. Kurze Darstellungen von Ordnungen. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994.

[Nei98] S. Neis. Reducing ideal arithmetic to linear algebra problems. In *Proc. of Algorithmic Number Theory Symposium III (ANTS III)*, volume 1423 of *Lecture Notes in Computer Science*. Springer, 1998.

[Nei02] S. Neis. *Zur Berechnung von Klassengruppen*. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2002.

[NR93] K. Nyberg and R. Rueppel. A new signature scheme based on the DSA giving message recovery. In *1st ACM Conference on Computer and Communications Security*, pages 58–61. ACM Press, 1993.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attack. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 427–437, 1990.

[Oka94] T. Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 1994.

[OO88] K. Ohta and T. Okamoto. Practical extension of Fiat-Shamir scheme. *Electronics Letters*, 24:955–956, 1988.

[OO90] K. Ohta and T. Okamoto. A modification of the Fiat-Shamir scheme. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 232–243. Springer, 1990.

[OW96] P.C. van Oorschot and M.J. Wiener. On Diffie-Hellman key agreement with short exponents. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 332–343. Springer, 1996.

[Pau96] S. Paulus. *Ein Algorithmus zur Berechnung der Klassengruppe quadratischer Ordnungen über Hauptidealringen*. PhD thesis, Universität GH Essen, Essen, Germany, 1996.

[Pfa03] T. Pfahler. *Das Eulerprodukt und der Gleichheitstest in der Idealklassengruppe algebraischer Zahlkörper*. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, Mai 2003. Preprint.

[PH78] S.C. Pohlig and M.E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Information Theory*, 24:106–110, 1978.

[Poh02] M.E. Pohst. *A User's Guide to KANT/KASH*. Universität Berlin, Germany, 2002. Available at `ftp://ftp.math.tu-berlin.de/pub/algebra/Kant/Kash`.

[Pol75] J.M. Pollard. A Monte Carlo method for factorization. *BIT*, 15:331–334, 1975.

[Pol78] J.M. Pollard. Monte Carlo methods for index computation $(mod\ p)$. *Math. Comp.*, 32:918–924, 1978.

[PS98] G. Poupard and J. Stern. Security analysis of a practical on the fly authentication and signature generation. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436. Springer, 1998.

[PW97] T. Papanikolaou and D. Weber. Personal communication. 1997.

[PZ89] M. Pohst and H. Zassenhaus. *Algorithmic algebraic number theory*. Cambridge University Press, Cambridge, 1989.

[Ros94] H.E. Rose. *A course in number theory*. Oxford University Press, Oxford, 1994.

[Sch82] R.J. Schoof. Quadratic fields and factorization. In H.W. Lenstra Jr. and R. Tijdeman, editors, *Computational methods in number theory*, pages 235–286. Mathematisch Centrum, 1982.

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.

[Sch90] C.P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1990.

[Sch91] H. Scheid. *Zahlentheorie*. BI Wissenschaftsverlag, Mannheim, 1991.

[Sch93] O. Schirokauer. Discrete logarithms and local units. *Phil. Trans. R. Soc. Lond. A*, 345:409–423, 1993.

[Sch94] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., Toronto, 1994.

[SE91] C.P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. In *FCT '91*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 1991.

[Sha71] D. Shanks. Class number, a theory of factorization and genera. In *Proc. Symp. Pure Math. 20*, pages 415–440. AMS, Providence, R.I., 1971.

[Sho98] V. Shoup. Why chosen ciphertext security matters. Research Report RZ 3076 (No. 93122), IBM Research Division, Zürich, Switzerland, 1998.

[Sho99] M.A. Shokrollahi. Relative class number of abelian number fields of prime conductor below 10000. *Math. Comp.*, 68:1717–1728, 1999.

[Sma99] N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12/3:193–196, 1999.

[Sma00] N.P. Smart. How secure are elliptic curves over composite extension fields? Technical report CSTR-00-017, Dept. of Computer Science, University of Bristol, 2000.

[Sta74] H.M. Stark. Some effective cases of the Brauer-Siegel Theorem. *Inventiones math.*, 23:135–152, 1974.

[Ste75] H.J. Stender. Eine Formel für Grundeinheiten in reinen algebraischen Zahlkörpern dritten, vierten und sechsten Grades. *Journal of Number Theory*, 7:235–250, 1975.

[Sti95] D.R. Stinson. *Cryptography: Theory and Practice*. CRC Press, Boca Raton, 1995.

[TA91] J. Tardo and K. Alagappan. SPX: Global authentication using public-key certificates. In *Proc. of the 1991 Symposium on Security and Privacy, IEEE*, pages 232–244, 1991.

[Tak01] T. Takagi. *New public-key cryptosystems with fast decryption*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2001. Available at `http://elib.tu-darmstadt.de/diss/000104/`.

[TAP90] J. Tardo, K. Alagappan, and R. Pitkin. Public-key based authentication using internet certificates. In *Proc. of USENIX Security II Workshop*, pages 121–123, 1990.

[Ter99] D.C. Terr. A modification of Shanks' baby-step-giant-step algorithm. *Mathematics of Computation*, 69:767–773, 1999.

[The00] P. Theobald. *Berechnung von Hermite-Normalformen*. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2000.

[Thi95] Christoph Thiel. *On the complexity of some problems in algorithmic algebraic number theory*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1995.

[van94] O. van Sprang. *Basisreduktionsalgorithmen für Gitter kleiner Dimension*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994.

[Vol00] U. Vollmer. Asymptotically fast discrete logarithms in quadratic number fields. In W. Bosma, editor, *Proc. of Algorithmic Number Theory Symposium IV (ANTS IV)*, volume 1838 of *Lecture Notes in Computer Science*, pages 581–594. Springer, 2000.

[Vor96] G.F. Voronoi. *On a generalization of the algorithm of continued fractions*. PhD thesis, Warsaw, 1896. Russian.

[Was97] L.C. Washington. *Introduction to cyclotomic fields*. Springer, New York, 1997.

[Web95] D. Weber. An implementation of the number field sieve to compute discrete logarithms mod p. In *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 95–105. Springer, 1995.

[Web96] D. Weber. Computing discrete logarithms with the general number field sieve. In *Proc. of Algorithmic Number Theory Symposium II (ANTS II)*, volume 1122 of *Lecture Notes in Computer Science*. Springer, 1996.

[Web97a] D. Weber. *On the computation of discrete logarithms in finite prime fields*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1997.

[Web97b] D. Weber. Personal communication. 1997.

[Wil90] H.C. Williams. The period length of Voronoi's algorithm for certain cubic orders. *Pub. Math. Debrecen*, 37:245–265, 1990.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *23rd IEEE Annual Symposium on the Foundations of Computer Science*, pages 80–91. IEEE Press, 1982.

Anhang A

Schranken für Regulatoren und Klassenzahlen von Stender-Körpern

In Kapitel 4.2 haben wir gesehen: NF-Kryptoverfahren benötigen

1. einen *kleinen Regulator*, um die effiziente Entscheidung zu ermöglichen, ob zwei gegebene Idealklassen gleich sind.

2. eine *große Klassenzahl*, um hinreichend sicher zu sein.

Für Stender-Körper haben wir experimentell bereits in Abschnitt 5.2.4 einen hinreichend kleinen Regulator und in Abschnitt 4.4.2.2 (Berechnungsergebnisse in Anhang B) eine hinreichend große Klassenzahl nachgewiesen.

Wir diskutieren nun theoretisch die Größe der Regulatoren und Klassenzahlen von Stender-Körpern $K = \mathbb{Q}(\omega)$ mit $\omega = \sqrt[n]{D^n \pm 1}$ vom Grad $n \in \{3, 4, 6\}$. Sei $D \in \mathbb{Z}_{>16}$, so dass $D^n \pm 1$ quadratfrei ist. Die in Tabelle 4.1 (Abschnitt 4.4.1) für kubische Stender-Körper angegebene obere Schranke für den Regulator und untere Schranke für die Klassenzahl stammen aus der Arbeit [Lou95]. Wir verfolgen einen anderen Ansatz, den wir analog auch für Stender-Körper der Grade $4, 6$ anwendeten. Damit weisen wir die Schranken aus Tabelle A.1 nach.

Grad	Obere Schranke für Regulator	Untere Schranke für Klassenzahl
$n = 3$	$R < 2\ln(2D)$	$h > \frac{c \cdot \sqrt{2}}{144\pi} \cdot \frac{D^3}{(\ln(2D))^2}$
$n = 4$	$R < 4 \cdot (\ln(\sqrt{3} \cdot D))^2$	$h > \frac{c}{64\pi} \cdot \frac{D^3}{(\ln D)^2}$
$n = 6$	$R < 9324 \cdot \ln(2D)$	$h > \frac{c}{35812448 \cdot \pi^2} \cdot \frac{D^{10}}{\ln(2D)}$

Tabelle A.1: Schranken für Regulator und Klassenzahl von Stender-Körpern, $d = 1, D > 16$

A.1 Stender-Körper dritten Grades

Sei nun $n = 3$, d.h. wir betrachten $K = \mathbb{Q}(\omega)$ mit $\omega = \sqrt[3]{D^3 \pm 1}$, $D \in \mathbb{Z}_{>16}$.

Der Körper K besitzt die Fundamentaleinheit $\xi_3 = \frac{\pm 1}{\omega - D}$ ([Ste75]). Die Konjugierten von ω sind

$$\omega^{(0)} = \omega, \;\; \omega^{(1)} = \varrho\omega, \;\; \omega^{(2)} = \varrho^2\omega,$$

wobei ϱ primitive dritte Einheitswurzel ist. Der Körper K besitzt folglich $r = 1$ reelle und $2s = 2$ komplexe Einbettungen in $\mathbb{C}$. Der Einheitenrang von K ist $r + s - 1 = 1$. Demnach ist

$$\begin{aligned} R &= |\det(\ln|\omega|)| = \ln|\xi_3| = \ln\left|\frac{1}{\omega - D}\right| && \text{(A.1)} \\ &< \begin{cases} 2\ln(2D) & \text{für } \omega = \sqrt[3]{D^3+1} \\ 2\ln(\sqrt{3}D) & \text{für } \omega = \sqrt[3]{D^3-1} \end{cases} && \text{(A.2)} \\ &\leqslant 2\ln(2D) && \text{(A.3)} \end{aligned}$$

Wir zeigen den Übergang von (A.1) nach (A.2). Sei $e \in \mathbb{R}$ so gewählt, dass

$$\sqrt[3]{D^3 + 1} = D + e \;\; . \tag{A.4}$$

Es gilt: $\sqrt[3]{D^3+1} > D$ und $\sqrt[3]{D^3+1} < D + 1$. (Letzteres ersieht man aus der binomischen Formel.) Es folgt

$$0 < e < 1 \;\; . \tag{A.5}$$

Durch Potenzieren von (A.4) mit 3 und Auswerten der binomischen Formel folgt die Existenz einer reellen Zahl e mit $0 < e < 1$, so dass

$$\begin{aligned} D^3 + 1 &= D^3 + 3D^2e + 3De^2 + e^3 \\ &= D^3 + e \cdot (3D^2 + 3De + e^2) \\ \Leftrightarrow \qquad 1 &= e \cdot (3D^2 + 3De + e^2) \end{aligned}$$

Mit (A.5) folgt:

$$1 < e \cdot (3D^2 + 3D + 1) \;\; . \tag{A.6}$$

Wegen $D \geqslant 4$ ist $D^2 > 3D + 1$, also folgt aus (A.6)

$$1 < e \cdot 4D^2 \;\; \Leftrightarrow \;\; \frac{1}{e} < (2D)^2$$

Einsetzen dieses Resultates in (A.1) liefert

$$R = \ln\left|\frac{1}{\omega - D}\right| = \ln\left|\frac{1}{e}\right| < \ln(2D)^2 = 2\ln(2D) \;\; , \tag{A.7}$$

also den ersten Teil der Behauptung (A.2). Zum Nachweis des zweiten Teils von (A.2) nimmt man die Existenz einer positiven reellen Zahl e mit $\sqrt[3]{D^3 - 1} = D - e$ an. Anschließend verfährt man analog.

$K = \mathbb{Q}(\sqrt[n]{D^n \pm 1})$	Quadratischer Teilkörper?	Normaler Körperturm?
$n = 3$	Nein	Nein
$n = 4$	Ja	Ja
$n = 6$	Ja	Nein

Tabelle A.2: Stender-Körpern mit quadratischem Teilkörper und normalem Körperturm

Wir haben in Abschnitt 4.4.1 bereits aufgezeigt, dass die Diskriminante Δ, die Klassenzahl h und der Regulator R eines algebraischen Zahlkörpers K vom Grad n über die *analytische Klassenzahlformel (Satz von Brauer–Siegel)* in Beziehung stehen, und zwar wie folgt (siehe [BS66]):

$$h = \frac{\kappa w \sqrt{|\Delta|}}{2^{r+s}\pi^s R} \ . \tag{A.8}$$

Für die Anzahl w der Einheitswurzeln in K gilt in unserem Fall $w = 2$ (siehe [Coh95, S. 208]). Die Anzahl der reellen Einbettungen von K in $\mathbb{C}$ ist $r = 1$, die Anzahl der komplexen Einbettungen von K ist $2s = 2$. Da $D^3 \pm 1$ nach Voraussetzung quadratfrei ist, gilt für die Diskriminante $\Delta \in \{-3(D^3 \pm 1)^2, -27(D^3 \pm 1)^2\}$, also $2D^6 < |\Delta| < 32D^6$. Wie oben nachgewiesen gilt $R < 2\ln(2D)$.

Für κ, den Wert der ζ-Funktion von K an der Stelle 1, gilt gemäß Stark [Sta74]:
Auf $\mathbb{Z}_{>0}$ sei die Funktion g definiert per $g(n) = 1$, falls $K/\mathbb{Q}$ einen normalen Körperturm besitzt, d.h. es existieren $m \in \mathbb{Z}_{>0}$ und Körper K_i mit $K_0 = \mathbb{Q} \leqslant K_1 \leqslant \ldots \leqslant K_m = K$, so dass K_i/K_{i-1} normal für $i \in \{1, \ldots, m\}$. Falls $K/\mathbb{Q}$ keinen normalen Körperturm besitzt, sei $g(n) = n!$. Falls K keinen quadratischen Teilkörper enthält, gilt

$$\kappa > \frac{c}{g(n) \cdot \ln|\Delta|} \tag{A.9}$$

Andernfalls gilt

$$\kappa > \frac{c}{n \cdot g(n) \cdot \sqrt[n]{|\Delta|}} \tag{A.10}$$

Dabei ist c jeweils eine effektiv berechenbare positive Konstante.

Um die geeignete Formel für κ gemäß Stark auszuwählen, müssen wir für die Stender-Körper jeweils entscheiden, ob K einen quadratischen Teilkörper bzw. einen normalen Körperturm über $\mathbb{Q}$ besitzt. Tabelle A.2 beantwortet diese Fragen.

Beispielsweise besitzt ein kubischer Stender-Körper ($n = 3$) keinen quadratischen Teilkörper. Nehmen wir an, es gäbe einen Körperturm $\mathbb{Q} \leqslant L \leqslant K$ mit Erweiterungsgrad $[L : \mathbb{Q}] = 2$. Nach der Gradformel gilt $3 = [K : \mathbb{Q}] = [K : L] \cdot [L : \mathbb{Q}] = [K : L] \cdot 2$, Widerspruch. Aus der Gradformel folgt auch, dass K keinen anderen echten Teilkörper als $\mathbb{Q}$ enthält. $K/\mathbb{Q}$ ist nicht normal, da K nicht Zerfällungskörper eines Polynoms aus $\mathbb{Q}[x]$ ist. Folglich besitzt $K/\mathbb{Q}$ auch keinen normalen Körperturm.

Hieraus ergibt sich die Auswahl der richtigen Formel für κ, und wir erhalten für eine effektiv berechenbare positive Konstante c

$$\kappa > \frac{c}{6\ln|\Delta|} > \frac{c}{6\ln(32D^6)} > \frac{c}{6\ln((2D)^6)} = \frac{c}{36\ln(2D)}$$

Einsetzen aller Teilresultate in $(\mathrm{A}.8)$ liefert die behauptete untere Schranke für h:

$$\begin{aligned} h &> \frac{\frac{c}{36\ln(2D)} \cdot 2 \cdot \sqrt{2D^6}}{2^2 \cdot \pi^1 \cdot 2\ln(2D)} \\ &= \frac{c \cdot \sqrt{2}}{144\pi} \cdot \frac{D^3}{(\ln(2D))^2} \end{aligned} \tag{A.11}$$

A.2 Stender-Körper vierten Grades

Für Stender-Körper vom Grade $n = 4$ gehen wir analog wie im Falle $n = 3$ vor. Wir skizzieren daher lediglich die Eckpunkte unserer Berechnungen. Wir betrachten

$$K = \mathbb{Q}(\omega), \quad \omega = \sqrt[4]{D^4 + 1} \ .$$

Gemäß Stender [Ste75] ist $\{\xi_2 = \frac{\omega+D}{\omega-D}, \xi_4 = \frac{1}{\omega-D}\}$ ein System von Fundamentaleinheiten. Wir haben hier $r = 2$, $s = 1$ und $w = 2$. Wir erhalten

$$\begin{aligned} R &= |\det \begin{pmatrix} \ln|\xi_2| & \ln|\xi_4| \\ \ln|\xi_2^{-1}| & \ln|-\xi_2^{-1} \cdot \xi_4| \end{pmatrix}| \\ &= \ln|\xi_2| \cdot |2\ln\xi_4 - \ln\xi_2| \\ &< 2\ln(\sqrt{2}D) \cdot 2\ln(\sqrt{3}D) \\ &< 4(\ln(\sqrt{3}D))^2 \end{aligned}$$

Mit Hilfe von Starks Formeln ergibt sich

$$\kappa > \frac{c}{32D^3}$$

für eine effektiv berechenbare positive Konstante c. Einsetzen der Teilresultate in $(\mathrm{A}.8)$ führt zur Behauptung

$$h > \frac{c}{64\pi} \cdot \frac{D^3}{(\ln D)^2} \, .$$

A.3 Stender-Körper sechsten Grades

Wir betrachten nun Stender-Körper vom Grade $n = 6$

$$K = \mathbb{Q}(\omega), \quad \omega = \sqrt[6]{D^6 + 1} \ .$$

Wir gehen wie in den Fällen $n = 3, 4$ vor. Stender [Ste75] zeigt, dass die Menge $\{\xi_2 = \frac{\omega+D}{\omega-D}, \xi_3 = \frac{\omega^3-D^3}{(\omega-D)^3}, \xi_6 = \frac{1}{\omega-D}\}$ ein System von Fundamentaleinheiten bildet. Wir haben hier $r = 2$, $s = 2$ und $w = 2$. Wir erhalten

$$\begin{aligned} R &= |\det \begin{pmatrix} \ln|\xi_2| & \ln|\xi_3| & \ln|\xi_6| \\ \ln|\xi_2^{-1}| & \ln|(\omega^3 + D^3) \cdot \xi_2^{-3} \cdot \xi_6^3| & \ln|\xi_2^{-1}\xi_6| \\ \ln\left|\frac{\xi_2\xi_3\xi_6^{-3}}{\omega^3+D^3}\right| & \ln\left|\frac{\xi_2^3\xi_6^{-3}}{\omega^3+D^3}\right| & \ln\left|\frac{\xi_2\xi_6^{-1}}{\omega^3+D^3}\right| \end{pmatrix}| \\ &< 9324 \cdot \ln(2D) \end{aligned}$$

Mit Hilfe von Starks Formeln ergibt sich

$$\kappa > \frac{c}{12963 \cdot D^5}$$

für eine effektiv berechenbare positive Konstante c. Einsetzen der Teilresultate in $(\mathrm{A}.8)$ führt zur behaupteten unteren Schranke

$$h > \frac{c}{35812448 \cdot \pi^2} \cdot \frac{D^{10}}{\ln(2D)} \,.$$

Anhang B

Klassenzahlberechnung für Stender-Körper

Aus unserer Analyse aus Kapitel 4.2 folgt: Damit NF-Kryptoverfahren hinreichend sicher sind, muss die Klassenzahl folgende Bedingungen erfüllen:

1. Sie muss hinreichend groß sein.
2. Sie muss hinreichend große Primteiler enthalten[1].

In den Abschnitten 4.4.1 und 4.4.2 haben wir diese Bedingungen für Stender-Körper bereits theoretisch untersucht; die experimentellen Resultate aus Abschnitt 4.4.2.2 basieren auf den Berechnungen, die wir jetzt beschreiben.

Mit Hilfe der PARI/GP-Implementierung [BBB+02] des Buchmann-Algorithmus [Buc89] haben wir die Klassenzahlen von Stender-Körpern $K = \mathbb{Q}(\omega)$ mit $\omega = \sqrt[n]{D^n - 1}$ der Grade $n \in \{3, 4, 6\}$ bestimmt ($D \in \mathbb{Z}_{>3}$). Da zur Klassenzahlberechnung kein effizienter Algorithmus bekannt ist, gelingt diese Berechnung nur für betragsmäßig kleine Diskriminanten. Die Tabellen B.1, B.2, B.3 zeigen einen Auszug unserer Berechnungsergebnisse. Die erste Spalte enthält jeweils den Parameter D des Körpers, die dritte Spalte dessen Diskriminante Δ. Die zweite Spalte zeigt die Laufzeit des Buchmann-Algorithmus in Millisekunden auf. In der vierten Spalte findet sich die Klassenzahl h, in der letzten Spalte jeweils deren Primfaktorzerlegung (PFZ).

Alle hier vorgestellten Laufzeiten wurden auf Sun-Solaris-Workstations mit UltraSparc II-Prozessoren und 450MHz Taktfrequenz gemessen. Wir haben bei unseren Experimenten als Stackgröße 900 MB realen physischen Speicher verwendet.

[1] Bei der Formulierung aus Abschnitt 4.2 impliziert die zweite Bedingung bereits die erste. Daher wird die erste Bedingung nicht explizit aufgeführt.

D	Zeit [ms]	Δ	h	PFZ(h)
100000000	227115240	$-27000000000000000000\ldots$	22876215012952840529280	$2^7 \cdot 3^5 \cdot 5 \cdot 19 \cdot 41 \cdot 557\ldots$
		$00540000000000000000000027$		$\cdot 339004348763$
101000000	33162160	$-1884355296924852071005\ldots$	872509132905029152128	$2^7 \cdot 3^6 \cdot 900971\ldots$
		$920817636686390532544 3787$		$\cdot 10378191289$
102000000	123610150	$-337848725779200000000\ldots$	10909388643314172055398	$2 \cdot 3^5\ldots$
		$00636724800000000000000003$		$\cdot 22447301735214345793$
103000000	120074670	$-3223941200628300000000\ldots$	33863632579481546571048	$2^3 \cdot 3^6 \cdot 11 \cdot 307\ldots$
		$00590072580000000000000027$		$\cdot 1719431851159357$
104000000	4845170	$-322651025974551419901 5\ldots$	64179151907758199712	$2^5 \cdot 3^4 \cdot 11104207\ldots$
		7219773937730027454547		$\cdot 2229828323$
105000000	59546210	$-4020286921875000000000\ldots$	20833968479283545002056	$2^3 \cdot 3^4 \cdot 83 \cdot 283\ldots$
		$00694575000000000000000003$		$\cdot 1368776275057273$
106000000	194054890	$-3830001603091200000000\ldots$	24531955075114399337400	$2^3 \cdot 3^4 \cdot 5^2 \cdot 2536273\ldots$
		$00643148640000000000000027$		$\cdot 597064359599$
107000000	62098650	$-5558260562403703703703\ldots$	1230821483920012917720	$2^3 \cdot 3^6 \cdot 5 \cdot 157\ldots$
		$71277809629629629629629 63$		$\cdot 268848672363331$
108000000	30773690	$-9715557079248979591836\ldots$	2544527586491656335720	$2^3 \cdot 3^8 \cdot 5 \cdot 29 \cdot 7114573\ldots$
		$75011892244897959183673 47$		$\cdot 46992689$
109000000	258012780	$-4528170299270700000000\ldots$	27062014909481186357286	$2 \cdot 3^7 \cdot 19 \cdot 47 \cdot 3299\ldots$
		$00699315660000000000000027$		$\cdot 2100136755727$
110000000	88216150	$-5314683000000000000000\ldots$	11470163822658367964730	$2 \cdot 3^5 \cdot 5 \cdot 23^2 \cdot 131\ldots$
		$00798600000000000000000003$		$\cdot 68113999131089$
111000000	56261090	$-5611243656483000000000\ldots$	26147536790056774474572	$2^2 \cdot 3^4 \cdot 1777 \cdot 5197\ldots$
		$00820578600000000000000003$		$\cdot 8738675806687$
112000000	123932900	$-5329321249996800000000\ldots$	52643592368705327051448	$2^3 \cdot 3^8 \cdot 137 \cdot 77093\ldots$
		$00758661120000000000000027$		$\cdot 94962027731$
113000000	91967810	$-6245855257827000000000\ldots$	11212668984559060940832	$2^5 \cdot 3^6 \cdot 173\ldots$
		$00865738200000000000000003$		$\cdot 2778339999900653$
114000000	62025020	$-6584917871808000000000\ldots$	23042537629861547596842	$2 \cdot 3^4 \cdot 1572017\ldots$
		$00888926400000000000000003$		$\cdot 90481137674773$
115000000	231273820	$-6245264067187500000000\ldots$	35216930684230809440976	$2^4 \cdot 3^6 \cdot 7 \cdot 9157 \cdot 20261\ldots$
		$00821272500000000000000027$		$\cdot 2324833531$
116000000	66478410	$-902369008450370370370 3\ldots$	1314768070461520540944	$2^4 \cdot 3^5 \cdot 13 \cdot 353\ldots$
		$71526589629629629629629 63$		$\cdot 2249981 \cdot 32751107$
117000000	100711910	$-769549260530700000000\ldots$	23556940734500616104064	$2^7 \cdot 3^4 \cdot 13\ldots$
		$00960967800000000000000003$		$\cdot 174775498089540421$
118000000	109690030	$-7288796213164800000000\ldots$	52265734616132492289552	$2^4 \cdot 3^9 \cdot 101 \cdot 277 \cdot 3359\ldots$
		$00887237280000000000000027$		$\cdot 1766016013$
119000000	63937170	$-8519282565843000000000\ldots$	23023344603873650102028	$2^2 \cdot 3^6 \cdot 19^2 \cdot 39551\ldots$
		$01011095400000000000000003$		$\cdot 552988657853$
120000000	53756520	$-8957952000000000000000\ldots$	26030587886692983711810	$2 \cdot 3^4 \cdot 5 \cdot 7 \cdot 11 \cdot 577\ldots$
		$01036800000000000000000003$		$\cdot 20794981 \cdot 34783549$

D	Zeit [ms]	Δ	h	PFZ(h)
121000000	200990970	$-8473756617146700000000\ldots$	43157407888333487359728	$2^4 \cdot 3^6 \cdot 13 \cdot 2281\ldots$
		$00956642940000000000000000027$		$\cdot 124778340020459$
122000000	95744800	$-9891911877312000000000\ldots$	18274138365026534236812	$2^2 \cdot 3^8 \cdot 29 \cdot 16361\ldots$
		$01089508800000000000000000003$		$\cdot 29147 \cdot 50350661$
123000000	305331590	$-1038847797506700000000\ldots$	23302006184484309948174	$2 \cdot 3^4\ldots$
		$00111652020000000000000000003$		$\cdot 143839544348668579927$
124000000	142234800	$-9815080708915200000000\ldots$	54735902786679345977544	$2^3 \cdot 3^4 \cdot 12473\ldots$
		$01029576960000000000000000027$		$\cdot 6772146699423761$
125000000	4279390	$-1321298094174985616970\ldots$	8674677689054698872	$2^3 \cdot 3^8 \cdot 546391\ldots$
		486787615735328703747		$\cdot 302475209$
126000000	95991820	$-1200451242412800000000\ldots$	35025701587214165258472	$2^3 \cdot 3^6 \cdot 139 \cdot 283\ldots$
		$00120022560000000000000000003$		$\cdot 53113 \cdot 2874532741$
127000000	172557150	$-1132885686966030000000\ldots$	53089013114232114620952	$2^3 \cdot 3^8 \cdot 13^2 \cdot 59877737\ldots$
		$00110612682000000000000000027$		$\cdot 99952243$
128000000	240688950	$-1319413953331200000000\ldots$	15572477122753685539644	$2^2 \cdot 3^9 \cdot 853 \cdot 4229503\ldots$
		$00125829120000000000000000003$		$\cdot 54823663$
129000000	132554540	$-1382482098816300000000\ldots$	28971604843104123785052	$2^2 \cdot 3^7 \cdot 13\ldots$
		$00128801340000000000000000003$		$\cdot 254753656599346873$
130000000	181663820	$-1303238430000000000000\ldots$	66051469392925419779760	$2^4 \cdot 3^5 \cdot 5 \cdot 29 \cdot 24793\ldots$
		$00118638000000000000000000027$		$\cdot 821039 \cdot 5755663$
131000000	102750896	$-3094232537314897959183\ldots$	2947553773942454482680	$2^3 \cdot 3^8 \cdot 5 \cdot 7 \cdot 2988019\ldots$
		67622215224489795918367347		$\cdot 536970359$
132000000	108648440	$-1586955840307200000000\ldots$	29586624507332131214304	$2^5 \cdot 3^6 \cdot 7 \cdot 3257\ldots$
		$00137998080000000000000000003$		$\cdot 55629110381657$
133000000	141150820	$-1494423230517630000000\ldots$	9387513098240665002624 0	$2^8 \cdot 3^6 \cdot 5 \cdot 2749\ldots$
		$00127042398000000000000000027$		$\cdot 75307 \cdot 485962739$
134000000	31602150	$-2382442987167078189300\ldots$	750715822100010589524	$2^2 \cdot 3^9 \cdot 13\ldots$
		4135029662551440329218107		$\cdot 733467598063939$
135000000	74159430	$-1816033542187500000000\ldots$	29788193902198949556840	$2^3 \cdot 3^6 \cdot 5 \cdot 137 \cdot 8243\ldots$
		$00147622500000000000000000003$		$\cdot 904587923539$
136000000	159465320	$-1708430099742720000000\ldots$	67731150771676739821608	$2^3 \cdot 3^7 \cdot 67 \cdot 1579\ldots$
		$00135834624000000000000000027$		$\cdot 36592556470511$

Tabelle B.1: Klassenzahlberechnung für Stender-Körper vom Grad 3

D	Zeit [ms]	Δ	h	PFZ(h)
362	599100	$-2025642674616899925561285783981 2$	2237020611256	$2^3 \cdot 1951 \cdot 2879 \cdot 49783$
363	424870	$-1340045256670285102269833099757568$	14142099343680	$2^6 \cdot 3^2 \cdot 5 \cdot 7^2 \cdot 11^2 \cdot 828209$
364	189660	$-2164096363187153146784579893965 2$	4562494382700	$2^2 \cdot 3^2 \cdot 5^2 \cdot 19 \cdot 173 \cdot 1033 \cdot 1493$
365	692330	$-1431378003147579197051917955680256$	15613256355840	$2^{12} \cdot 3^3 \cdot 5 \cdot 11 \cdot 13 \cdot 197453$
366	435340	$-2311175984718675027042723287 2$	5739733633536	$2^9 \cdot 3^4 \cdot 7 \cdot 17 \cdot 31 \cdot 37517$
367	771000	$-1528384893224869923100664248076288$	6105859072000	$2^{13} \cdot 5^3 \cdot 41 \cdot 145433$
368	271050	$-2467367361053888875000978132 1732$	2382115061920	$2^5 \cdot 5 \cdot 11 \cdot 37 \cdot 593 \cdot 61687$
369	1227090	$-1631384603637026801948093865330688$	20160777781248	$2^{15} \cdot 3 \cdot 13 \cdot 15775849$
370	201610	$-2633180802757511646893009932 0004$	12518154035712	$2^9 \cdot 3^5 \cdot 13 \cdot 67 \cdot 71 \cdot 1627$
371	568850	$-1740711829745318993684625432823808$	8068161708032	$2^{13} \cdot 131 \cdot 7518191$
372	220240	$-2809152200284735975350381640 3972$	7099648618944	$2^6 \cdot 3 \cdot 7 \cdot 83 \cdot 1093 \cdot 58229$
373	1386000	$-1856717997313408762735523989833728$	4887030816000	$2^8 \cdot 3^2 \cdot 5^3 \cdot 29^2 \cdot 20177$
374	233350	$-2995844157457904196815107099 0532$	9409013916804	$2^2 \cdot 3^2 \cdot 53^2 \cdot 4373 \cdot 21277$
375	671010	$-1979772001805236816421437500000256$	26644600065024	$2^{10} \cdot 3^2 \cdot 31 \cdot 127 \cdot 734347$
376	470540	$-3193847168650076029839794613 4532$	2734502365568	$2^7 \cdot 31 \cdot 41 \cdot 53 \cdot 103 \cdot 3079$
377	455460	$-2110260976111968702797609316681728$	14411208187904	$2^{13} \cdot 11 \cdot 29 \cdot 107 \cdot 51539$
378	203930	$-3403780841112199014249847713 1972$	9591805638912	$2^8 \cdot 3 \cdot 7^2 \cdot 61 \cdot 4178431$
379	1366640	$-2248591087639954334225884754167808$	6278593727232	$2^8 \cdot 3^3 \cdot 811 \cdot 1120051$
380	323740	$-3626295163919733239044541632 0004$	7684684416720	$2^4 \cdot 3^4 \cdot 5 \cdot 47 \cdot 353 \cdot 71479$
381	4266280	$-2395188365716974290198388725106688$	12347002060800	$2^{18} \cdot 3 \cdot 5^2 \cdot 11 \cdot 37 \cdot 1543$
382	232120	$-3862071825054572840780597651 3732$	3416226582528	$2^{12} \cdot 3^3 \cdot 83 \cdot 372173$
383	1477220	$-2550499560299915582344098404108288$	13570685829120	$2^{15} \cdot 3 \cdot 5 \cdot 7 \cdot 79 \cdot 49927$
384	1293530	$-4111825578178959966109665381 5812$	7963396875776	$2^9 \cdot 83 \cdot 97 \cdot 941 \cdot 2053$
385	635920	$-2714993032993444865476240259680256$	21237215232000	$2^{15} \cdot 3 \cdot 5^3 \cdot 23 \cdot 163 \cdot 461$
386	257170	$-4376305660699816231375719016 3652$	4448146726296	$2^3 \cdot 3^3 \cdot 7 \cdot 13 \cdot 23 \cdot 61 \cdot 101 \cdot 1597$
387	641780	$-2889159681416237574572956117485568$	25808613408768	$2^{18} \cdot 3 \cdot 223 \cdot 147163$
388	311030	$-4656297264764536195899689048 7812$	3421567260400	$2^4 \cdot 5^2 \cdot 701 \cdot 3167 \cdot 3853$
389	960740	$-3073513897978886177248410678806528$	7896260740096	$2^{10} \cdot 17 \cdot 23 \cdot 103 \cdot 191473$
390	230590	$-4952623062872811159685481292 0004$	21833723759400	$2^3 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 157 \cdot 11037167$
391	711710	$-3268594564165758768520076147591168$	11442795970560	$2^{17} \cdot 3 \cdot 5 \cdot 233 \cdot 24979$
392	223080	$-5266144789833494016176355912 9092$	4898407652064	$2^5 \cdot 3^4 \cdot 17 \cdot 19 \cdot 53 \cdot 101 \cdot 1093$
393	992310	$-3474966081441819809397441454901248$	19202014511104	$2^{16} \cdot 41 \cdot 1607 \cdot 4447$
394	605250	$-5597764882840891402064402565 5492$	3076989956616	$2^3 \cdot 3 \cdot 31 \cdot 83 \cdot 1399 \cdot 35617$
395	1197510	$-3693219439934767229330726838880256$	14669692668928	$2^{10} \cdot 11 \cdot 23^2 \cdot 2461913$
396	453720	$-5948428181491192605772778240 7172$	9923593961216	$2^8 \cdot 13 \cdot 17 \cdot 43 \cdot 89 \cdot 45833$
398	352420	$-6319123689607030392712765610 9252$	8250840494336	$2^8 \cdot 11^2 \cdot 29^2 \cdot 367 \cdot 863$
399	38690	$-1726717081736298682878545 92$	14847082496	$2^{15} \cdot 53 \cdot 83 \cdot 103$
400	211750	$-6710886400786432000030720000 0004$	14430081361920	$2^{11} \cdot 3 \cdot 5 \cdot 7 \cdot 29 \cdot 599 \cdot 3863$
401	55116150	$-4425602836733530372236909628 6197248$	6157491404800	$2^{15} \cdot 5^2 \cdot 23 \cdot 281 \cdot 1163$
402	258040	$-7124799189641674120996422489 0052$	9490549572864	$2^8 \cdot 3^3 \cdot 179 \cdot 7670693$
403	2695190	$-4697864854170411018882747674 052608$	9981551878976	$2^6 \cdot 17 \cdot 61 \cdot 151 \cdot 643 \cdot 1549$
404	303890	$-7561994770743818968349522442 9572$	9324760838400	$2^8 \cdot 3^2 \cdot 5^2 \cdot 7^2 \cdot 37 \cdot 89293$

D	Zeit [ms]	Δ	h	PFZ(h)
405	815690	−4985402692860437923136277226080256	46648522902720	$2^6 \cdot 3 \cdot 5 \cdot 79 \cdot 1249 \cdot 492467$
406	233330	−802365772733999486071716599850 92	6471314242560	$2^{10} \cdot 3^3 \cdot 5 \cdot 17 \cdot 277 \cdot 9941$
407	629130	−5288991585274774101723777980008448	15049071378432	$2^{14} \cdot 3^4 \cdot 7 \cdot 13 \cdot 29 \cdot 4297$
408	381150	−8511026611962809756328148839 6292	10383422988288	$2^{13} \cdot 3 \cdot 7 \cdot 19 \cdot 43 \cdot 73877$
410	193580	−9025396121104665459928429132 0004	16267643106144	$2^5 \cdot 3 \cdot 401 \cdot 1117 \cdot 378317$
411	662190	−5947601028310840831661815946364928	22610388331776	$2^8 \cdot 3 \cdot 7 \cdot 1879 \cdot 2238319$
412	266550	−9568119346184189705652659411 6612	6293059920720	$2^4 \cdot 3^3 \cdot 5 \cdot 37 \cdot 78741991$
413	705300	−6304353908651630824209537332922368	14634369057792	$2^{10} \cdot 3 \cdot 11 \cdot 61 \cdot 7099541$
414	259190	−1014061010308753898236605277 38052	15209712291906	$2 \cdot 3^4 \cdot 11^2 \cdot 71 \cdot 10928543$
415	628570	−6680625484342007945299554794080256	31678654279680	$2^{12} \cdot 3^4 \cdot 5 \cdot 19096411$
416	291320	−1074434534262396298932113196974 12	7732618005504	$2^{10} \cdot 3 \cdot 7 \cdot 23 \cdot 73 \cdot 79 \cdot 2711$
417	491940	−7077381802185944859750732122687488	22162740469760	$2^{13} \cdot 5 \cdot 11 \cdot 89 \cdot 131 \cdot 4219$
418	1043480	−1138086764429277831924149414 28932	6137514058240	$2^9 \cdot 5 \cdot 17 \cdot 163 \cdot 281 \cdot 3079$
419	2158640	−7495631729838942353473024536332288	10121982835776	$2^6 \cdot 3 \cdot 7 \cdot 179 \cdot 263 \cdot 159977$
420	309920	−1205178779581779085400726035 20004	59849222620896	$2^5 \cdot 3^4 \cdot 23 \cdot 59 \cdot 131 \cdot 193 \cdot 673$
421	849610	−7936428634757072385668397193054208	29825110149120	$2^{11} \cdot 3^2 \cdot 5 \cdot 11 \cdot 13 \cdot 2263099$
422	347730	−1275878746096525007710759268 08772	5894522050200	$2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 23^2 \cdot 73 \cdot 36343$
423	197060	−1005839503893304916305662791 68	78436761600	$2^{17} \cdot 3 \cdot 5^2 \cdot 79 \cdot 101$
424	758870	−1350362193822263087113819728 60932	5008185664928	$2^5 \cdot 31 \cdot 107 \cdot 137 \cdot 479 \cdot 719$
425	2301350	−8890109821828229021431306300000256	74464685457408	$2^{21} \cdot 3^2 \cdot 89 \cdot 97 \cdot 457$
426	268270	−1428812301297793635489494980 24132	56111830034592	$2^5 \cdot 3 \cdot 29 \cdot 59 \cdot 10753 \cdot 31769$
427	2532880	−9405339260991518921596176399484928	14849377337344	$2^{15} \cdot 11 \cdot 317 \cdot 129959$
428	277620	−1511420190590281606371755289671 72	8218646539456	$2^6 \cdot 1787 \cdot 2083 \cdot 34499$
429	725220	−9947809765813745268401378130577408	35130416058624	$2^8 \cdot 3 \cdot 17 \cdot 41 \cdot 43 \cdot 379 \cdot 4027$
430	486570	−1598385232030761459375314561 20004	11234378963190	$2 \cdot 3 \cdot 5 \cdot 37 \cdot 109 \cdot 1609 \cdot 57709$
431	565380	−1051882445251231670306912895 0745088	14480193110016	$2^{14} \cdot 3 \cdot 83 \cdot 3549401$
432	212440	−1689915359130252418602231626 46532	55755283723776	$2^9 \cdot 3^5 \cdot 7 \cdot 13 \cdot 4924571$
433	1190860	−1111974227440017049611684795 2857088	17744875487232	$2^{24} \cdot 3 \cdot 41 \cdot 8599$
434	326360	−1786227393966632825584671167 58212	7832335271712	$2^5 \cdot 3 \cdot 7 \cdot 457 \cdot 827 \cdot 30839$
435	584530	−1175198013857823955328869788 5280256	376809750331392	$2^{16} \cdot 3 \cdot 7 \cdot 23 \cdot 11904059$
436	564560	−1887547383339280938037526911 25252	6328219824220	$2^2 \cdot 5 \cdot 7 \cdot 83 \cdot 1667 \cdot 326693$
437	713120	−1241701509250876669787990085 9680768	80657621581824	$2^{17} \cdot 3^2 \cdot 7 \cdot 31 \cdot 487 \cdot 647$
438	1339860	−1994110945995477271221590044 51012	14594092738560	$2^{11} \cdot 3^3 \cdot 5 \cdot 37 \cdot 67 \cdot 107 \cdot 199$
439	1199180	−1311638658243770908577033794 1424128	15407558885376	$2^{16} \cdot 3^3 \cdot 7 \cdot 503 \cdot 2473$
440	250300	−2106163631239682232947089715 20004	28857488793600	$2^{13} \cdot 3 \cdot 5^2 \cdot 17^2 \cdot 331 \cdot 491$
441	2395750	−1385169878568045829154560456 3437568	30058709696512	$2^{14} \cdot 7 \cdot 19 \cdot 43 \cdot 149 \cdot 2153$
442	230120	−2223961289244054196984065209 89892	17527389886330	$2 \cdot 5 \cdot 17 \cdot 149 \cdot 691961701$
443	1022620	−1462462301883193971831694439 4242048	15988975430400	$2^8 \cdot 3^5 \cdot 5^2 \cdot 7 \cdot 23 \cdot 63857$
444	13970	−2810994185158174759265710852	70904882176	$2^{10} \cdot 19^2 \cdot 59 \cdot 3251$
445	754540	−1543690022400986041071080307 5680256	41742789713088	$2^6 \cdot 3 \cdot 31 \cdot 277 \cdot 1151 \cdot 21997$
446	4298850	−2477868734945745405845081148 40772	6466822092232	$2^3 \cdot 109 \cdot 3967 \cdot 1869443$
447	1012870	−1629034353528845907993337809 2075008	83490378088448	$2^{18} \cdot 331 \cdot 449 \cdot 2143$

D	Zeit [ms]	Δ	h	PFZ(h)
448	316930	−261454523050571790918047601000452	10661786204160	$2^{11} \cdot 3 \cdot 5 \cdot 7 \cdot 13 \cdot 23 \cdot 79 \cdot 2099$
449	1174850	−171868409275295882817717783035924 48	35996999270400	$2^{14} \cdot 3^4 \cdot 5^2 \cdot 7 \cdot 23^2 \cdot 293$
450	388040	−27581009423990471296924207500000 4	99703568605824	$2^7 \cdot 3^2 \cdot 29 \cdot 83 \cdot 35956891$
451	1003620	−18128357949868338433331684071465 1648	19721729920000	$2^{10} \cdot 5^4 \cdot 17^2 \cdot 106627$
452	656040	−29088492125652913508729886087885 2	6859367235840	$2^8 \cdot 3 \cdot 5 \cdot 7 \cdot 23 \cdot 193 \cdot 57487$
454	238290	−30671161701772978704559176711597 2	13491710319648	$2^5 \cdot 3 \cdot 11 \cdot 2999 \cdot 4260167$
455	647850	−20154717964761056499483434781280256	71907493968384	$2^9 \cdot 3 \cdot 2389 \cdot 19595971$
456	349850	−32332411934124821603360770469069 2	20464759979008	$2^{10} \cdot 2579 \cdot 7749173$
457	1864400	−21243905760022212989115695384987648	19784369307648	$2^{17} \cdot 3 \cdot 11 \cdot 67 \cdot 233 \cdot 293$
458	689110	−34075773857498652334612203141549 2	8869620632040	$2^3 \cdot 3^2 \cdot 5 \cdot 7^5 \cdot 17 \cdot 53 \cdot 1627$
459	1047840	−22386808888021289110184243611322368	191377183116288	$2^{10} \cdot 3 \cdot 13^2 \cdot 41 \cdot 71 \cdot 126631$
460	346000	−35904920671827807394286049472000 4	23024247375360	$2^9 \cdot 3 \cdot 5 \cdot 7 \cdot 11^2 \cdot 19 \cdot 311 \cdot 599$
461	908140	−23585824898999258284142485107427328	50155539474432	$2^{11} \cdot 3 \cdot 7 \cdot 251 \cdot 4646179$
462	1083470	−37823672807458465704552800103341 2	27094491112000	$2^6 \cdot 5^3 \cdot 13 \cdot 19^2 \cdot 271 \cdot 2663$
464	888550	−39836003147636330363539718589645 2	8763959522208	$2^5 \cdot 3 \cdot 11 \cdot 131 \cdot 63352703$
465	587210	−26162268594139412525006089296480256	129789229793280	$2^{20} \cdot 3 \cdot 5 \cdot 37 \cdot 83 \cdot 2687$
466	374400	−41946042412374530010115675587501 2	12637565855298	$2 \cdot 3^2 \cdot 877 \cdot 1249 \cdot 640957$
470	490390	−46476593246436785695869876172000 4	31547071592448	$2^{11} \cdot 3 \cdot 29 \cdot 199 \cdot 889727$
4000	102476930	−6710886400 . . .	111206 . . .	$2^5 \cdot 3^2 \cdot 17 \cdot 15359 \ldots$
		00007864320000000030720000000000 4	92422162508128	·147886142077
4001	384717510	−430786992940 . . .	50207 . . .	$2^{20} \cdot 3^4 \cdot 7^2 \cdot 37 \cdot 131 \ldots$
		04045571772803126460422881771540 48	28236178407424	·248893
4002	89352670	−67512626 . . .	43150 . . .	$2 \cdot 3 \cdot 7^2 \cdot 11 \cdot 1409 \cdot 4013 \ldots$
		32782839197692212235521491449192 3652	67604464452374	·235975783
4003	45647260	−4333781851625 . . .	72729 . . .	$2^{13} \cdot 3^2 \cdot 17 \cdot 103 \cdot 1163 \cdot 3929 \ldots$
		05719569329853569613486767497420 8	09697405722624	·12329
4004	74180220	−67918614350	41241 . . .	$2^8 \cdot 7 \cdot 10267 \cdot 84751 \ldots$
		24697270758198123344659523001037 2	81690341925632	·2644913
4005	227104110	−4359836574637	265308 . . .	$2^8 \cdot 3 \cdot 7 \cdot 127 \cdot 257 \cdot 63103 \ldots$
		05809462084227704517086753648025 6	54105006811392	·2396101
4006	1162044490	−68326839215	23761 . . .	$2 \cdot 3^2 \cdot 7^2 \cdot 13 \cdot 29 \cdot 184463 \ldots$
		21125150976270021900912485094829 2	09517310109854	·38738897

Tabelle B.2: Klassenzahlberechnung für Stender-Körper vom Grad 4

D	Zeit [ms]	Δ	h	PFZ(h)
20	994790	$-782757850848954231029789\ldots$	4968439007463360	$2^6 \cdot 3^8 \cdot 5 \cdot 11 \cdot 13^2 \cdot 17 \cdot 103 \cdot 727$
		859840233280000729		
21	542350	$-267301472236310148231799\ldots$	21122368533161280	$2^6 \cdot 3^8 \cdot 5 \cdot 7^2 \cdot 11^2 \cdot 47 \cdot 79 \cdot 457$
		8438652709891500032		
22	18217010	$-136586561538501349775498\ldots$	13013228295547200	$2^6 \cdot 3^4 \cdot 5^2 \cdot 7 \cdot 47 \cdot 97^2 \cdot 163 \cdot 199$
		49090416797949465625		
23	4025090	$-331697804923285773196095\ldots$	144848398311751680	$2^{19} \cdot 3^{10} \cdot 5 \cdot 7^2 \cdot 13^2 \cdot 113$
		8935611375333254400000		
24	4901280	$-229392794540196141003599\ldots$	23367812373255168	$2^{10} \cdot 3^{10} \cdot 7^2 \cdot 13 \cdot 19 \cdot 37 \cdot 863$
		0518976447130107913		
25	19761830	$-404676300763640096783638\ldots$	325288176416686080	$2^{15} \cdot 3^7 \cdot 5 \cdot 13 \cdot 17 \cdot 19 \cdot 149 \cdot 1451$
		27857971248359375046656		
26	2634480	$-205082203223044962000831\ldots$	583050699312979968	$2^{13} \cdot 3^{10} \cdot 7 \cdot 19 \cdot 29 \cdot 47 \cdot 61 \cdot 109$
		3301903929111386236953		
27	439588040	$-502730708010031338168075\ldots$	267229554127683840	$2^8 \cdot 3^{11} \cdot 5 \cdot 7 \cdot 11^2 \cdot 41 \cdot 33937$
		3256334065070342400000		
28	2741450	$-189438598946141826978379\ldots$	1534041665028642816	$2^{11} \cdot 3^9 \cdot 67 \cdot 167 \cdot 227 \cdot 14983$
		$5840636566805910446 5625$		
29	405212780	$-347414101889792829438126\ldots$	1772239759560972288	$2^{11} \cdot 3^8 \cdot 17^2 \cdot 31^2 \cdot 474899$
		$1435852873405721119598592$		
30	6028910	$-185302020156117017651284\ldots$	1125090245805020928	$2^8 \cdot 3^8 \cdot 97 \cdot 101 \cdot 349 \cdot 409 \cdot 479$
		4057829690032805000009		
31	72789060	$-256896578299886819047351\ldots$	5526788146661488896	$2^8 \cdot 3^{10} \cdot 7^3 \cdot 41 \cdot 53 \cdot 139 \cdot 3529$
		$48710419304982277000251392$		
32	2518420	$-665896686609715716449338\ldots$	24927161265457920	$2^8 \cdot 3^6 \cdot 5 \cdot 7^3 \cdot 3191 \cdot 24407$
		70288529834602060625		
33	31740160	$-206938206802093619495218\ldots$	6344909399284558848	$2^{10} \cdot 3^9 \cdot 59 \cdot 61 \cdot 1783 \cdot 49057$
		$3119010518892056403200000$		
34	7860880	$-641349914224698669463342\ldots$	87761196309943296 00	$2^{12} \cdot 3^8 \cdot 5^2 \cdot 7^2 \cdot 31 \cdot 967 \cdot 8893$
		$7578423023937778408073753$		
35	16741930	$-979374594486160627075136\ldots$	84452474948163033600	$2^9 \cdot 3^{10} \cdot 5^2 \cdot 7^3 \cdot 151 \cdot 2157341$
		$3746134479780896118550466 56$		
36	32318440	$-439863102836554915163158\ldots$	8113665855451910400	$2^8 \cdot 3^7 \cdot 5^2 \cdot 191 \cdot 41047 \cdot 73939$
		$07855849845506399415 9113$		
37	562989860	$-518751332354377607712726\ldots$	88244684888594909184	$2^{11} \cdot 3^{11} \cdot 31 \cdot 83 \cdot 239 \cdot 395537$
		$7892098637115559792185600000$		
38	1172130	$-747380735497824731354980\ldots$	9307954540995840	$2^8 \cdot 3^6 \cdot 5 \cdot 31 \cdot 103 \cdot 109 \cdot 28661$
		3545099607086365625		
39	160151960	$-310713909074294744237046\ldots$	131555800011551956992	$2^{13} \cdot 3^7 \cdot 7^3 \cdot 17 \cdot 37 \cdot 43 \cdot 181 \cdot 4373$
		$7491708399270395508953272 32$		

D	Zeit [ms]	Δ	h	PFZ(h)
40	19734930	$-840479777884367736115306\ldots$ $1055277459046549299200007 29$	153384721856385610752	$2^{10}\cdot 3^4\cdot 7\cdot 29\cdot 47\cdot 137\cdot 9839\cdot 143791$
41	3344380	$-189685973140816553407670\ldots$ 068658499886783383552	12241396136180736	$2^{10}\cdot 3^{12}\cdot 7^2\cdot 103\cdot 4457$
42	59682780	$-448457427946954421134676\ldots$ $422773958020830851898156 25$	161559155307653222400	$2^{12}\cdot 3^9\cdot 5^2\cdot 23\cdot 43\cdot 73\cdot 521\cdot 2131$
43	19403900	$-301398539855873575281968\ldots$ $803346903279365215091200 00$	3273983468673408000	$2^{10}\cdot 3^9\cdot 5^3\cdot 3727\cdot 348671$
44	35792350	$-146658697353715492214062\ldots$ $645004350789886582633401 10553$	673962288838815744000	$2^{15}\cdot 3^{10}\cdot 5^3\cdot 9973\cdot 279407$
45	143553690	$-227403893891974203434809\ldots$ $216009395590259346960950 00576$	596395366686819779328	$2^8\cdot 3^{14}\cdot 173\cdot 7937\cdot 354727$
46	44471540	$-556496553490068825224701\ldots$ $962217395202747462540417 63353$	633305539506682224000	$2^7\cdot 3^5\cdot 5^3\cdot 7\cdot 13\cdot 19\cdot 250703\cdot 375779$

Tabelle B.3: Klassenzahlberechnung für Stender-Körper vom Grad 6

Index